How to improve branching algorithms for deciding on grammars equivalence

Didier Caucal

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France

Tél. (1) 39 63 55 11

Rapports de Recherche

N° 618

HOW TO IMPROVE BRANCHING ALGORITHMS FOR DECIDING ON GRAMMARS EQUIVALENCE

Didier CAUCAL

Février 1987

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

How to improve branching algorithms for deciding on grammars equivalence (*)

Didier CAUCAL

IRISA , Campus de Beaulieu , F35042 Rennes-Cédex , FRANCE

Publication interne n° **334**

48 pages

Janvier 1987

Abstract : We present two algorithms in order to decide on the equivalence of simple grammars [Korenjak-Hopcroft] and of stateless deterministic pushdown automata with acceptance on pushdown letters [Oyamaguchi-Honda]. These two algorithms are of complexities polynomial in time and space in the valuation and the description length of the compared grammars, and exponential if we only consider the last parameter.

Comment améliorer des algorithmes de branchement décidant de l'équivalence de grammaires

Résumé : on présente deux algorithmes pour décider de l'équivalence des grammaires simples [Korenjak-Hopcroft] et des automates à pile déterministes sans état avec acceptation sur lettres de pile [Oyamaguchi-Honda]. Ces deux algorithmes sont de complexités, en temps et en espace, polynomiales selon la valuation et la longueur de description des grammaires (ou automates) comparées, et exponentielle si l'on ne tient compte que du dernier paramètre.

(*) Submitted to publication

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

1. Introduction

The problem of equivalence for two classes $\mathbf{C}$ and $\mathbf{C}'$ (for one class if $\mathbf{C} = \mathbf{C}'$) of context-free grammars (respectively of pushdown automata) is: can we decide for any couple (G,G') of $\mathbf{C}\times\mathbf{C}'$, whether the language generated (respectively recognized) by G starting from its axiom is equal to the one generated by G'? The equivalence is undecidable for the context-free class. For the last twenty years [Gi-Gr 66] , the problem has been unsolved for the class of deterministic pushdown automata (in short, dpda). On the other hand, there is a lot of algorithms, the so-called equivalence, to decide on the equality of deterministic context-free languages subclasses. We distinguish two families of equivalence algorithms. A first one is the family of the so-called Valiant algorithms (cf [Va 74], [Va-Pa 75],[Oy-Ho 78], [Oy-Ho-In 80]). To compare any couple of automata, these algorithms build a third automaton which simulates the joint action of the other two and, for instance, will not accept no word if and only if the languages accepted by the first two are equal. The other family of equivalence algorithms is the one of the so called branching or K-H algorithms (among them [Ko-Ho 66], [Ol-Pn 77], [Ha-Ha-Ye 79], [Co 83a], [To 84]). For any couple of grammars (or automata) to be compared, these algorithms build a finite tree whose root is the pair of axioms and each label is a pair of non-terminal letters. Although the method of branching algorithm is direct and isolate the nature of the considered equivalence, Valiant's method is used more frequently.

For a context-free grammar G (or a pushdown automaton), let n_G denote the description length of G and ℓ_G its valuation, namely the smallest number such that for any non-terminal letter A (or in the case of an automaton, for any couple (A,q), A being a pushdown letter and q a state) generating a non-empty language $\mathbf{L}$, there exists in $\mathbf{L}$ a word of length less than or equal to ℓ_G. The time and space complexity of an equivalence algorithm is at least exponential with respect to n_G and ℓ_G ($G = G_1 \cup G_2$ for (G_1,G_2) to be considered) and generally a double exponential with respect to n_G only (generally for the classes to be compared, the valuation depends exponentially on the description length). It is such a result we would like to reconsider.

We present two branching algorithms. The first one is an improvement on the K-H algorithm [Ko-Ho 66], that is the algorithm which decides on the simple grammars

equivalence with complexity polynomial in time and space as regard to n_G and ℓ_G , and not exponential. The second one is an extension of the first one and decides on the stateless dpda equivalence with acceptance on pushdown letters. Its complexity (as regard on n_G and ℓ_G) is also polynomial in time and space and not exponential [Oy-Ho 78]. Depending only on the description length, the complexities of these algorithms are only exponential instead of double exponential. Apart from the question of complexity, it is already surprising to be able to solve the second problem with a branching algorithm (the appropriate equivalence is not a congruence). These two algorithms are the issue of a general method which resumes the work of [Ha-Ha-Ye 79] [Co 81 a,b] [Ca 86] and will be presented latter on.

This article does not require any particular knowledge. The second paragraph sums up briefly the notions of languages, trees, Thue congruence and context-free grammars. The third paragraph is concerned with the simple grammars equivalence. The paragraph 4 provides an algorithm of equivalence for simple grammars together with examples. The paragraph 5 proves the validity of the algorithm and evaluates its complexity. The paragraphs 6, 7, 8 deal with the equivalence of stateless dpda in the same way as paragraphs 3, 4, 5.

2. Thue congruence and context-free grammar

Let Σ^* be the set of finite words formed by concatenation from an alphabet, finite and nonempty set of symbols or letters, Σ. The empty word is denoted by ε and the symbol for concatenation is the dot (which can be omitted). Let $\mathbb{N}$ (resp. $\mathbb{N}_+$) be the set of non negative integers (resp. and non nul). We define $[0] = \varnothing$ and $\forall i \in \mathbb{N}_+$, $[i] = \{1, \ldots, i\}$. For $i \in \mathbb{N}$ and $u \in \Sigma^*$, we denote by $|u|$ the length (total number of occurences of letters) of word u, by $u(i)$ the i^{th} letter of u, by $u|i$ the prefix or left factor of u with length $\min(|u|,i)$ and by $u \backslash i$ the suffix of u from its i^{th} letter with $u(i)=u\backslash i=\varepsilon$ for $i \notin [|u|]$. For any $u \in \Sigma^*$ and $i \in \{0\} \cup [|u|]$, $u|(-i) = u|(|u|-i)$ is the left factor of u obtained by omitting the last i letters of u. We define a partial pre-order on Σ^*, denoted by $<_{FG}$ and called strict left factor, as follows : $\forall u,v \in \Sigma^*$, $u <_{FG} v$ if $u \neq v$ and $\forall i \in [|u|]$, $u(i)=v(i)$. From $<_\Sigma$ a total pre-order on Σ, we define the total pre-orders $<_{lex}$ and $<_{par}$ on Σ^*, respectively called lexicographic pre-order and parallel pre-order, as follows : $\forall u,v \in \Sigma^*$, $u <_{lex} v$ if $u <_{FG} v$ or if there exists $i \in [\min(|u|,|v|)]$ such that $u|(i-1) = v|(i-1)$ and $u(i) <_\Sigma v(i)$; and $u <_{par} v$ if ($|u| < |v|$) or ($|u| = |v|$ and $u <_{lex} v$). A language over Σ is a subset of Σ^*. We say that a language L is a prefix language when the relation $<_{FG}$ restricted to L is empty. $FG(L)$ is the set of left factors of words of L. For any set E, we denote by $\#E$ the cardinality of E and by $P(E)$ (resp. $P_f(E)$) the set of subsets (resp. finite subsets) of E.

For any binary relation R, we define $Dom(R) = \{x / \exists y\ (x,y) \in R\}$ the domain of R and $Im(R) = R(Dom(R))$ the range of R with $R(A) = \{y / (x,y) \in R$ and $x \in A\}$ the image of the set A by the relation R ; $R^{-1} = \{(y,x) / (x,y) \in R\}$ is the reverse relation of R. ι_E is the identity relation on E. For any $i \in \mathbb{N}_+$, $R^i = \{(x,y) / (x,z) \in R$ and $(z,y) \in R^{i-1}\}$ the i times composed relation of R with $R^0 = \iota_E$; $R^* = \bigcup_{i \in \mathbb{N}} R^i$ the reflexive and transitive closure of R.

A tree T over a set E is a partial function of $\mathbb{N}_+$ into E such that the domain

Dom(T) of the graph of T, the set of **nodes** of T, is closed under prefix (if $uv \in$ Dom(T) then $u \in$ Dom(T)) and the "left brothers" of a node exist (for any $i \in \mathbb{N}_+$, if $ui \in$ Dom(T) then $\forall j \in [i]$, $uj \in$ Dom(T)). The range Im(T) of the graph of T is the set of **labels** of T. We say that a tree is finite if its domain is finite. We denote by $\|T\| = \#Dom(T)$ the **size** of T and $d(T) = \max\{|u| / u \in Dom(T)\}$ the **depth** of T. $Fl(T) = \{u \in Dom(T) / u.1 \notin Dom(T)\}$ is the set of terminal nodes or **leaves** of T. We denote by T\u the **subtree** of T at node u, that is the tree with domain $Dom(T\backslash u) = \{v / uv \in Dom(T)\}$ and such that for any $v \in Dom(T\backslash u)$, $(T\backslash u)(v) = T(uv)$.

A **semi-Thue system** over Σ is a binary relation R on Σ^*. R is closed under concatenation if R is closed under left and right concatenation, that is for any $(u,v) \in R$ and $w \in \Sigma^*$, we have $(wu,wv) \in R$ and $(uw,vw) \in R$. R is a **congruence** if R is an equivalence relation closed under concatenation. We set $\xleftrightarrow{}_R = \xrightarrow{}_R \cup (\xrightarrow{}_R)^{-1}$; $\xrightarrow{*}_R = (\xrightarrow{}_R)^*$ the **derivation** operation according to R and $\xleftrightarrow{*}_R = (\xleftrightarrow{}_R)^*$ the smallest congruence containing R and so-called the **Thue congruence** generated by R. For any $n \in \mathbb{N}$, we define $\xrightarrow{n}_R = (\xrightarrow{}_R)^n$ and $\xleftrightarrow{n}_R = (\xleftrightarrow{}_R)^n$. A word u of Σ^* is **irreducible** according to R if $\xrightarrow{}_R(\{u\})$ is empty. R is **noetherian** if no word of Σ^* has an infinite derivation; this implies that any word derives to an irreducible word. And R is **confluent** if it satisfies the following property : $\forall u,v,w \in \Sigma^* / w \xrightarrow{*}_R u$ and $w \xrightarrow{*}_R v$, $\exists w' \in \Sigma^* / u \xrightarrow{*}_R w'$ and $v \xrightarrow{*}_R w'$. R is **canonical** if R is noetherian and confluent, i.e. any word u of Σ^* derives (according to R) to one and only one irreducible word, denoted by $u{\downarrow}R$ and called **normal form** of u according to R.

A **context-free grammar** over Σ with axiom S is a finite relation G in $N \times (\Sigma \cup N)^*$ with $S \in N = Dom(G)$ a set of symbols disjoint of Σ. An element of N (resp. Σ) is called a non-terminal letter (resp. terminal letter) of G. A word of N^* (resp. Σ^*) is called a **non-terminal** (resp. **terminal**) of G. We denote by $\|G\| = \#Dom(G)$ the size of G, $|G| = \max\{|\alpha| / \alpha \in Im(G)\}$ the length of G and $\Sigma_G = \{\alpha(i) \in \Sigma / \alpha \in Im(G)$ and $i \in [|\alpha|]\}$ the set of terminal letters used by G. G being a semi-Thue system over $(\Sigma \cup N)^*$, we define for any

$U \in (\Sigma \cup N)^*$, $L(G,U) = \{u \in \Sigma^* / U \xrightarrow[G]{*} u\}$ the language of terminals which derive from U according to G. We say that a language L over Σ is a context-free language if there is a context-free grammar G over Σ with axiom S such that $L = L(G,S)$. We define the mapping τ_G which with any $U \in (\Sigma \cup N)^*$ associates the smallest length $\tau_G(U)$ of the words of $L(G,U)$ if that language is nonempty and otherwise $\tau_G(U) = \infty$. This convention makes τ_G a homomorphism $((\Sigma \cup N)^*,.)$ to $(\mathbb{N} \cup \{\infty\},+)$: $\forall\ U,V \in (\Sigma \cup N)^*$, $\tau_G(UV) = \tau_G(U) + \tau_G(V)$. We use the notation $N_1 = \{A \in N / L(G,A) \neq \emptyset\}$ and $N_0 = N - N_1$. N_1 and τ_G on N are calculable in $O(\#G.|G|)$. The valuation ℓ_G of G is $\max \{\tau_G(A) / A \in N_1\}$ and we note $\ell_G \leq |G|^{\|G\|+1}$. We define a total pre-order $<_N$ on N, increasing with τ_G, and Val an application of N_1 into Σ^* which with any $A \in N_1$ associates Val(A) a word of $L(G,A)$ of minimal length : $|Val(A)| = \tau_G(A)$. We can define such an application Val in $O(\#G.|G| + \|G\|.\ell_G)$. We say that G is in Greibach Normal Form, in short GNF, when $Im(G) \subset \Sigma.N^*$. G is reduced (resp. ε-free; prefix) if for any $A \in N$, $L(G,A) \neq \emptyset$ i.e. $N_0 = \emptyset$ (resp. $\varepsilon \notin L(G,A)$; $L(G,A)$ is prefix). We denote by $\xrightarrow[G]{*}_g = (\xrightarrow[G]{}_g)^*$ the left derivation according to G with $\xrightarrow[G]{}_g$ the relation on $(\Sigma \cup N)^*$ defined as follows : $\forall\ U,V \in (\Sigma \cup N)^*$, $U \xrightarrow[G]{}_g V$ if and only if $\exists\ u \in \Sigma^*$, $\exists\ (A,\alpha) \in G$ such that $U = uAU'$ and $V = u\alpha U'$. For any $U \in (\Sigma \cup N)^*$ and $u \in \Sigma^*$, we define $R_G(U,u) = \{V\}$ if $U = u\,V$ and else

$$R_G(U,u) = \{V \in (\Sigma \cup N)^* / U \xrightarrow[G]{*}_g vU' \xrightarrow[G]{}_g uV \text{ with } v <_{FG} u \text{ and } U'(1) \in N\}.$$

3. Equivalence for the family of simple grammars

A simple grammar G is a context-free grammar in GNF such that : for any non-terminal letter A and terminal letter a, if $(A,a\alpha),(A,a\beta) \in G$ then $\alpha=\beta$ (i.e. $\#R_G(A,a) \leq 1$; so, a simple grammar is a LL(1) grammar in GNF). A simple language is a language generated by a simple grammar (from its axiom). It is also a language no reduced to the empty word and recognized by a stateless dpda with acceptance on pushdown letters..

The equivalence problem for the class of simple grammars is that of deciding whether $L(G_1,S_1)=L(G_2,S_2)$, for any two simple grammars G_1 and G_2 with respective axioms S_1 and S_2. This problem is inter-reducible decide whether $L(G,\alpha)=L(G,\alpha')$, for any simple grammar G and for any $\alpha,\alpha' \in (Dom(G))^*$. In fact, for the only if part, we can, provided a possible renaming, assume that $Dom(G_1)$ and $Dom(G_2)$ are disjoints and then we take $G = G_1 \cup G_2$ with $\alpha=S_1$ and $\alpha'=S_2$. For the if part, excluding the trivial case where α or α' is the empty word, we take $G_1 = G \cup \{(S_1,a\beta)/\ a\in\Sigma$ and $\beta \in R_G(\alpha,a)\}$ and $G_2 = G \cup \{(S_2,a\beta)/\ a\in\Sigma$ and $\beta \in R_G(\alpha',a)\}$ with $S_1,S_2 \notin Dom(G)$.

We define the equivalence relation $\equiv_G$ for any grammar G over Σ, by :

$\forall\ U,V \in (\Sigma \cup Dom(G))^*$, $U \equiv_G V$ iff $L(G,U)=L(G,V)$. It is a congruence on $(\Sigma \cup Dom(G))^*$. Furthermore, the equivalence problem for the family of simple grammars corresponds to the decidability of $\equiv_G$ restricted to $(Dom(G))^*$, for any simple grammar G.

For any grammar G, we know how to determine $N_1=\{A\in Dom(G)/\ \tau_G(A)<\infty\}$, how to compute τ_G on N_1 and how to build $\overline{G} = \{(A,\alpha)\in G/\ \tau_G(\alpha)<\infty\}$ in $O(\#G.|G|)$. Moreover, $Dom(\overline{G})=N_1$ and $\forall\ \alpha,\beta \in (Dom(G))^*$, we have $(\alpha \equiv_G \beta) \Leftrightarrow ((\alpha,\beta \in N_1^*$ and $\alpha \equiv_{\overline{G}} \beta)$ or $(\alpha,\beta \notin N_1^*))$.

For any simple grammar G, $\overline{G}$ is a reduced simple grammar. We are going to restrict the study of $\equiv_G$ on $(Dom(G))^*$ to the reduced simple grammars.

From now on, G is a reduced and simple grammar over Σ and $N=Dom(G)$.

G being in GNF, we have $\forall\ \alpha \in N^*$, $\forall\ u \in \Sigma^*$, $R_G(\alpha,u) = \{\beta\in N^*/\ \alpha \xrightarrow[G]{*} u\beta\ \}$

G being ε-free, we have $R_G(\alpha,u) = \bigcup_{\alpha'\in X} R_G(\alpha'.\alpha\backslash 2,u\backslash 2)$ where $X = R_G(\alpha(1),u(1))\}$

Since we have $\forall\ A\in N$, $\forall\ a\in\Sigma$, $\#R_G(A,a) \leq 1$, we obtain $\forall\ \alpha\in N^*$, $\forall\ u \in \Sigma^*$, $\#R_G(\alpha,u) \leq 1$

Therefore, for any $\alpha \in N^*$, $L(G,\alpha)$ is prefix so G is a prefix grammar. In addition, G being

reduced, we have for $\alpha \in N^*$ and $u \in \Sigma^*$, $R_G(\alpha,u) \neq \emptyset \Leftrightarrow u \in FG(L(G,\alpha))$ and in that case, $\{v/\ uv \in L(G,\alpha)\} = L(G,\alpha')$ where $\{\alpha'\}=R_G(\alpha,u)$. Then, for any $\alpha,\beta \in N^*$, $\alpha \equiv_G \beta$ iff for each $a \in \Sigma$, $(R_G(\alpha,a) \neq \emptyset \Leftrightarrow R_G(\beta,a) \neq \emptyset)$ and $\alpha' \equiv_G \beta'$ where $\{\alpha'\}=R_G(\alpha,a)$ and $\{\beta'\}=R_G(\beta,a)$.

This is the most natural way to work with an equivalence : to carry over the analysis of the equivalence of one pair to the pairs obtained by (left) direct derivation or precisely, with the mapping [Ha 78], denoted by T_A and called **left parallel transformation**, of $N^*\times N^*$ into $P_f(N^*\times N^*)$, defined as follows :

Definition 3.1 :

$$\forall\ \alpha,\beta \in N^*,\ T_A(\alpha,\beta) = \begin{cases} \{(\varepsilon,\varepsilon)\} \text{ if } \alpha=\beta=\varepsilon \\ \emptyset \text{ if for } \{\alpha',\beta'\}=\{\alpha,\beta\},\ \exists\ a \in \Sigma / R_G(\alpha',a) = \emptyset \neq R_G(\beta',a) \\ \{(\alpha',\beta') / \exists\ a\in\Sigma\ ,\ \{\alpha'\}=R_G(\alpha,a),\ \{\beta'\}=R_G(\beta,a)\} \text{ else} \end{cases}$$

The previous property, called validity of T_A, is equivalent to :

$$\forall\ \alpha,\beta \in N^*\ ,\ \alpha \equiv_G \beta \Leftrightarrow \emptyset \neq T_A(\alpha,\beta) \subset \equiv_G$$

The following step consists in applying again the transformation T_A to the set of pairs obtained by T_A, and so on. We extend T_A on $P_f(N^*\times N^*)$ by : $\forall$ E finite $\subset N^*\times N^*$, $T_A(E) = \emptyset$ if $\exists\ (\alpha,\beta) \in E$ such that $T_A(\alpha,\beta)=\emptyset$, else $T_A(E) = \bigcup_{(\alpha,\beta)\in E} T_A(\alpha,\beta)$. For any integer n, we denote by T_A^n the application composing T_A n times. We shall write $T_A^n(\alpha,\beta)$ instead of $T_A^n(\{(\alpha,\beta)\})$. The study of the equivalence of a pair in applying iteratively T_A corresponds to :

Proposition 3.1 :

Let G be a reduced and simple grammar over Σ and $\alpha,\beta \in (Dom(G))^*$

The following properties are equivalent :

a) $\alpha \equiv_G \beta$ i.e. $L(G,\alpha) = L(G,\beta)$

b) $\forall\ u \in \Sigma^*,\ R_G(\alpha,u) \neq \emptyset \Leftrightarrow R_G(\beta,u) \neq \emptyset$

c) $\forall\ n \geq 0,\ T_A^n(\alpha,\beta) \neq \emptyset$

Proof : a) $\Rightarrow$ b) since G is reduced

b) $\Rightarrow$ a) since G is simple, hence in GNF and prefix

To show that b) $\Leftrightarrow$ c), we shall prove by induction on $n \in \mathbb{N}$ the following lemma

lemma 3.1 :

$\forall n \in \mathbb{N}$, $\forall \alpha,\beta \in N^*$,
$T_A^n(\alpha,\beta) \neq \emptyset \Leftrightarrow (\forall u \in \Sigma^* / |u| \leq n, R_G(\alpha,u) \neq \emptyset \Leftrightarrow R_G(\beta,u) \neq \emptyset)$
and in that case,
$T_A^n(\alpha,\beta) = \{(\alpha',\beta') / \{\alpha'\} = R_G(\alpha,u), \{\beta'\} = R_G(\beta,u), u \in \Sigma^n\} \cup \{(\varepsilon,\varepsilon) / \tau_G(\alpha) < n\}$

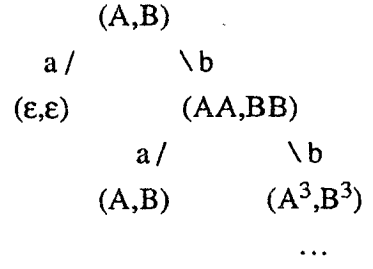
that completes the proof of the proposition 3.1 □

Thus, for any $\alpha,\beta \in N^*$, $\alpha \not\equiv_G \beta \Leftrightarrow \exists n \geq 1 / T_A^n(\alpha,\beta) = \emptyset$. This provides a semi-decision procedure of the inequivalence. For any $\alpha,\beta \in N^*$, we define $Divg(\alpha,\beta)$ the smallest integer n such that $T_A^n(\alpha,\beta) = \emptyset$ if $\alpha \not\equiv_G \beta$ and if $\alpha \equiv_G \beta$ then $Divg(\alpha,\beta) = \infty$. Divg is called the left divergence operator. One way to ensure termination is to determine a bound $b_{\alpha,\beta}$ of $Divg(\alpha,\beta)$ depending on the description length of G and on $|\alpha|$ and $|\beta|$. As a result, we could have $\alpha \equiv_G \beta \Leftrightarrow T_A^{b_{\alpha,\beta}}(\alpha,\beta) \neq \emptyset$. The decision algorithm stated in the following paragraph, allows the determination of such a bound, which is minimal.

To decide on the equivalence, a first method consists in stopping the transformation of one pair if it is reflexive or has (or its symmetry) already been transformed. This method, although valid, is insufficient to obtain a break in the development. It suffices to consider the following example :

Example : let G $\quad A \rightarrow a + bAA$
$\quad B \rightarrow a + bBB$

We represent the successive transformations applied from the pair (A,B) by a tree :



The transformation T_A being insufficient, we introduce a new transformation, denoted generally by T_B and called cutting transformation, which is also an application of $N^* \times N^*$ into $P_f(N^* \times N^*)$. Then, we define an algorithm, called branching algorithm, which for any $(\alpha,\beta) \in$

$N^* \times N^*$ builds through transformations T_A and T_B a tree $T_{\alpha,\beta}$ with root (α,β), called **equivalence tree**. A pair is **final** if it is reflexive or has (or its symmetry) already been transformed in the tree. A pair is **non-transformable** if its transformation (by T_A or T_B) gives the empty set. We stop the building of the tree as soon as a non-transformable pair appears or if all the labels of the leaves of the tree are final pairs. We say that the algorithm is **valid** if for any pair (α,β) of non-terminals, we have $\alpha \not\equiv_G \beta$ if, and only if, the building of $T_{\alpha,\beta}$ blocks on a non-transformable pair. Moreover, if the algorithm is with a finite termination (always terminates) i.e. $\forall\, \alpha,\beta \in N^*$, $T_{\alpha,\beta}$ is finite, then the algorithm is a decision algorithm of $\equiv_G$ on N^*. In this case, such an algorithm is an **equivalence algorithm** for $\equiv_G$. We note that no constraint has been imposed on the order in which the pairs of the tree are developped. Furthermore, the choice of using T_A or T_B to transform a pair is free. In order to get a valid algorithm with finite termination, it suffices that T_B satisfy the two conditions, which T_A satisfies :

- the validity condition : $\forall\, (\alpha,\beta) \in \mathrm{Dom}(T_B),\ \alpha \equiv_G \beta \Leftrightarrow \emptyset \neq T_B(\alpha,\beta) \subset \equiv_G$
- the monotony condition : $\forall\, (\alpha,\beta) \in \mathrm{Dom}(T_B) / T_B(\alpha,\beta) \neq \emptyset$ if $\alpha \not\equiv_G \beta$ then
  $\exists\, (\alpha',\beta') \in T_B(\alpha,\beta) / \mathrm{Divg}(\alpha',\beta') < \mathrm{Divg}(\alpha,\beta)$

Before defining the transformation T_B, we must establish the link between the congruence $\equiv_G$ on N^* and $\overset{*}{\underset{R}{\longleftrightarrow}}$ the smallest congruence on N^* containing a binary relation R on N^*. *Why ? Deciding $\equiv_G$ on N^* amounts to exhibiting a finite system generating $\equiv_G$, that is to prove the existence of a finite relation R on N^* such that $\equiv_G = \overset{*}{\underset{R}{\longleftrightarrow}}$! And also, optimizing the branching algorithm corresponds to extracting a minimal (for the inclusion) generating system of $\equiv_G$!* These results have their origine in [Co 83 a,b].

To prove that a binary relation S on N^* is included in $\equiv_G$, it suffices to prove that S is **closed by the transformation** T_A, that is $\emptyset \neq T_A(S) \subset S$. In fact, if $\emptyset \neq T_A(S) \subset S$ then by induction on $n \in \mathbb{N}$, we have $\forall\, n \in \mathbb{N}$, $\emptyset \neq T_A^{\,n}(S) \subset S$ hence, by proposition 1, $S \subset \equiv_G$. In order to obtain an equivalence relation, we have a more general condition, as follows : $\emptyset \neq T_A(S) \subset \overset{*}{\underset{S}{\longleftrightarrow}}$. In this case, we say that S is **self-proving** [Co 83 a,b].

Proposition 3.2 :

Let R be a binary relation on N^*
$\overset{*}{\underset{R}{\longleftrightarrow}}$ is closed by $T_A \Leftrightarrow$ R is self-proving

Proof : If $\overset{*}{\underset{R}{\longleftrightarrow}}$ is closed under T_A then $\emptyset \neq T_A(\overset{*}{\underset{R}{\longleftrightarrow}}) \subset \overset{*}{\underset{R}{\longleftrightarrow}}$

in particular $\emptyset \neq T_A(R) \subset \overset{*}{\underset{R}{\longleftrightarrow}}$ i.e. R is self-proving

Conversely, if R is self-proving : $\emptyset \neq T_A(R) \subset \overset{*}{\underset{R}{\longleftrightarrow}}$

then by lemma 3.2 below, we have

$\emptyset \neq T_A(\overset{*}{\underset{R}{\longleftrightarrow}}) \subset \overset{*}{\underset{S}{\longleftrightarrow}} = \overset{*}{\underset{R}{\longleftrightarrow}}$ where $S = R \cup \overset{*}{\underset{R}{\longleftrightarrow}}$

i.e. $\overset{*}{\underset{R}{\longleftrightarrow}}$ is closed by T_A transformation □

Lemma 3.2 :

Let R be a binary relation on N^* satisfying $T_A(R) \neq \emptyset$

Then $\emptyset \neq T_A(\overset{*}{\underset{R}{\longleftrightarrow}}) \subset \overset{*}{\underset{S}{\longleftrightarrow}}$ with $S = R \cup T_A(R)$

Proof : Let us show by induction on $n \in \mathbb{N}$, for $T_A(R) \neq \emptyset$ and $S = R \cup T_A(R)$, that

$\emptyset \neq T_A(\overset{n}{\underset{R}{\longleftrightarrow}}) \subset \overset{*}{\underset{S}{\longleftrightarrow}}$

- for n=0, we have $\emptyset \neq T_A(\iota_{N^*}) = T_A(\overset{n}{\underset{R}{\longleftrightarrow}}) \subset \iota_{N^*} \subset \overset{*}{\underset{S}{\longleftrightarrow}}$
- for n=1, let $(\alpha,\beta) \in \underset{R}{\longleftrightarrow}$. By symmetry, we may suppose $\alpha \underset{R}{\longrightarrow} \beta$

  i.e. $\exists \lambda,\mu \in N^*, \exists (\alpha_0,\beta_0) \in R / \alpha = \lambda\alpha_0\mu$ and $\beta = \lambda\beta_0\mu$

  if $\lambda \neq \varepsilon$ then $\emptyset \neq T_A(\alpha,\beta) \subset \underset{R}{\longleftrightarrow}$

  if $\lambda = \varepsilon$ then $T_A(R) \neq \emptyset \Rightarrow T_A(\alpha,\beta) \neq \emptyset$ and $T_A(\alpha,\beta) \subset \underset{S}{\longleftrightarrow}$

  thus, $\emptyset \neq T_A(\underset{R}{\longleftrightarrow}) \subset \underset{S}{\longleftrightarrow} \subset \overset{*}{\underset{S}{\longleftrightarrow}}$

- suppose that the property is true for all integer between 1 and $n \geq 1$ and consider $(\alpha,\beta) \in \overset{m}{\underset{R}{\longleftrightarrow}}$ with $m = n+1$

  so $\exists \gamma \in N^* / (\alpha,\gamma) \in \underset{R}{\longleftrightarrow}$ and $(\gamma,\beta) \in \overset{n}{\underset{R}{\longleftrightarrow}}$

  by induction hypothesis, $T_A(\alpha,\gamma)$ and $T_A(\gamma,\beta)$ are nonempty and included in $\overset{*}{\underset{S}{\longleftrightarrow}}$

  from lemma 3.1, $\forall a \in \Sigma, R_G(\alpha,a) \neq \emptyset \Leftrightarrow R_G(\gamma,a) \neq \emptyset \Leftrightarrow R_G(\beta,a) \neq \emptyset$

  then $T_A(\alpha,\beta) \neq \emptyset$ and

  $T_A(\alpha,\beta) = \{(\alpha',\beta')/ \{\alpha'\}=R_G(\alpha,a) , \{\beta'\}=R_G(\beta,a), a \in \Sigma\} \cup \{(\varepsilon,\varepsilon)/ \text{ if } \alpha=\varepsilon\}$

  hence $\forall (\alpha',\beta') \in T_A(\alpha,\beta), (\alpha',\beta') \in \overset{*}{\underset{S}{\longleftrightarrow}}$

  thus $\emptyset \neq T_A(\overset{m}{\underset{R}{\longleftrightarrow}}) \subset \overset{*}{\underset{S}{\longleftrightarrow}}$

this completes the induction and also the proof of lemma 3.2 □

Note that a self-proving relation is included in $\equiv_G$. Moreover, for a finite relation R, $\overset{*}{\underset{R}{\longleftrightarrow}}$ is semi-decidable, hence it is semi-decidable whether a finite relation is self-proving. Therefore, if $\equiv_G$ is finitely generated then $\equiv_G$ is decidable. This is the application of the theorem 5.8.1 in [Co 83 b] to the simple grammars.

Theorem 1 :

Let G be a simple grammar over Σ and N = Dom(G)
If there exists a finite binary relation R on N^* / $\equiv_G = \overset{*}{\underset{R}{\longleftrightarrow}}$ on N^*
then $\equiv_G$ is decidable on N^*

Proof : If $\exists\ R \subset N^* \times N^*$ / R is finite and $\equiv_G \cap (N^* \times N^*) = \overset{*}{\underset{R}{\longleftrightarrow}} \cap (N^* \times N^*)$
then $\forall\ \alpha,\beta \in N^*, \alpha \equiv_G \beta \Leftrightarrow \exists\ R_{\alpha,\beta}$ finite and self-proving / $(\alpha,\beta) \in R_{\alpha,\beta}$

In fact, by validity of T_A, $\equiv_G$ is closed by T_A so $\overset{*}{\underset{R}{\longleftrightarrow}}$ is closed by T_A and by proposition 3.2, $R_{\alpha,\beta} = R \cup \{(\alpha,\beta)\}$ is finite and self-proving. The converse comes from the fact a self-proving relation can only contain equivalent pairs.

To decide on the equivalence of two non-terminals α and β, we activate simultaneously two algorithms. The first one is the semi-decision algorithm of the inequivalence, that is for its n^{th} activation, it tests if $T_A^n(\alpha,\beta) = \emptyset$ and in the affirmative, we have $\alpha \not\equiv_G \beta$. The second one is the semi-decision algorithm of $\equiv_G$, using two others algorithms.One, called Range, gives for an integer i, a finite binary relation Range(i) on N^* such that $\{Range(i)/i \in \mathbb{N}_+\} = P_f(N^* \times N^*)$. The other one, called Selfproof, is a semi-decision algorithm that a finite binary relation on N^* is self-proving. Let f the bijection of $\mathbb{N}^2$ on $\mathbb{N}$ defined by $f(i,j) = (i+j)(i+j+1)/2 + j$. For its n^{th} activation, the semi-decision algorithm of $\equiv_G$ tests for $(i,j) = f^{-1}(n)$ if Range(i) contains the pair (α,β) and is self-proving, with the procedure Selfproof, after j "computing steps"; in the positive case, we have $\alpha \equiv_G \beta$.

The formal description of the algorithm is given in [Ca 85] . □

Conversely, the branching algorithm that we mention later, enables us to determine a finite system generating the equivalence. For that, we define the cutting transformation T_B such that :

$\forall\ (\alpha,\beta) \in Dom(T_B)$, if $\alpha \equiv_G \beta$ then $\emptyset \neq T_B(\alpha,\beta) \subset \equiv_G$
and $(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}}$ for $R = T_B(\alpha,\beta) \neq \emptyset$

Henceforth the cut pairs (transformed by T_B) will be excluded from the generating system. This property valids T_B but is not enough for T_B to be monotonous. First of all, let us show some properties of the Divg operator.

Lemma 3.3 :

$\forall\ \alpha, \beta, \gamma, \delta \in N^*$ / $\gamma \equiv_G \delta$

a) $1 \leq Divg(\alpha,\beta) \leq Divg(\alpha\gamma,\beta\delta) \leq Divg(\alpha,\beta) + \tau_G(\gamma) = Divg(\gamma\alpha,\delta\beta)$

b) $\min(Divg(\alpha,\beta),Divg(\beta,\gamma)) \leq Divg(\alpha,\gamma)$

Proof :

i) Let us prove $Divg(\alpha,\beta) \leq Divg(\alpha\gamma,\beta\delta)$

either $\alpha\gamma \equiv_G \beta\delta$ i.e. $Divg(\alpha\gamma,\beta\delta) = \infty$ hence the inequality

or $\alpha\gamma \not\equiv_G \beta\delta$. Let $n = Divg(\alpha\gamma,\beta\delta) \geq 1$

from lemma 3.1 and by the symmetry between α and β, it suffices to assume that

$$\exists\, u \in \Sigma^n \,/\, R_G(\alpha\gamma,u) \neq \emptyset = R_G(\beta\delta,u)$$

Then, we have the following two cases :

- either $R_G(\alpha,u) \neq \emptyset$

  but $R_G(\beta\delta,u) = \emptyset$ so $R_G(\beta,u) = \emptyset$

  hence $Divg(\alpha,\beta) \leq |u| = n = Divg(\alpha\gamma,\beta\delta)$

- or $R_G(\alpha,u) = \emptyset$

  then $\exists\, u',u'' \in \Sigma^* \,/\, u'u'' = u$ and $\{\varepsilon\} = R_G(\alpha,u')$ thus $R_G(\gamma,u'') \neq \emptyset$

  but $\gamma \equiv_G \delta$ and G is reduced, so $R_G(\delta,u'') \neq \emptyset$

  since $R_G(\beta\delta,u'u'') = \emptyset$, we obtain $R_G(\beta,u') \neq \{\varepsilon\}$

  . if $R_G(\beta,u') = \emptyset$ then $Divg(\alpha,\beta) \leq |u'| < |u| = Divg(\alpha\gamma,\beta\delta)$

  . if $R_G(\beta,u') \neq \emptyset$. Let $\{\beta'\} = R_G(\beta,u')$. $\beta' \neq \varepsilon$ and G is ε-free so

  $\exists\, a \in \Sigma \,/\, R_G(\beta,u'a) \neq \emptyset$; but $R_G(\alpha,u'a) = R_G(\varepsilon,a) = \emptyset$

  thus $Divg(\alpha,\beta) \leq |u'| + 1 \leq |u| = Divg(\alpha\gamma,\beta\delta)$

this proves i)

ii) The others inequalities are shown in the same way (we must take account that G is prefix); this completes the proof of lemma 3.3 □

Remark : lemma 3.3 a) implies that $\equiv_G$ is simplifiable :

$$\text{if } \lambda\alpha \equiv_G \lambda\beta \text{ or } \alpha\lambda \equiv_G \beta\lambda \text{ then } \alpha \equiv_G \beta$$

From the properties of Divg, we deduce that for any (α,β) belonging to the smallest congruence generated by R, there exists a pair of R which the divergence is less or equal to the one of (α,β).

Proposition 3.3 :

Let R be a nonempty binary relation on N^*

$$\forall\, (\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}},\ \exists\, (\alpha_0,\beta_0) \in R \,/\, Divg(\alpha_0,\beta_0) \leq Divg(\alpha,\beta)$$

Proof : by induction on the length of the derivation and by using lemma 3.3 □

If T_B satisfies the previous property, T_B is valid but non-monotonous : we obtain a large inequality which cannot be strict (just take the identity for T_B). However, the cutting operation T_B has been introduced in order to obtain a branching algorithm with finite termination. But, T_B must be defined as "to decrease" the pairs of the tree. In addition, T_B must satisfy the following condition :

$\exists\ \mu_G$ application of $N^* \times N^*$ into $\mathbb{N}$ such that

$$\forall\ (\alpha,\beta) \in \mathrm{Dom}(T_B)\ ,\ \forall\ (\alpha',\beta') \in T_B(\alpha,\beta)\ ,\ \mu_G(\alpha',\beta') < \mu_G(\alpha,\beta)$$

This additional condition suffices to validate any branching algorithm if it always terminate.

Proposition 3.4 :

If T_B satisfies the following properties for all pairs $(\alpha,\beta) \in \mathrm{Dom}(T_B)$:

$$\alpha \equiv_G \beta \Rightarrow \emptyset \neq T_B(\alpha,\beta) \subset \equiv_G$$

$$(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}} \text{ if } R = T_B(\alpha,\beta) \neq \emptyset$$

$$\forall\ (\alpha',\beta') \in T_B(\alpha,\beta),\ \mu_G(\alpha',\beta') < \mu_G(\alpha,\beta) \text{ for } \mu_G \text{ application of } N^* \times N^* \text{ into } \mathbb{N}\ ,\ \text{depending only on G}$$

then any branching algorithm with a finite termination is valid, and therefore is an equivalence algorithm for $\equiv_G$. Furthermore, for any (α,β) of $\equiv_G$ and for $T_{\alpha,\beta}$ the finite tree obtained from (α,β) with a branching algorithm, the set $R_{\alpha,\beta}$ of labels of $T_{\alpha,\beta}$ having been transformed by T_A, is a finite and self-proving system and $(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}}$

Proof : Let $\alpha,\beta \in N^*$ and $T_{\alpha,\beta}$ be a finite tree obtained from a branching algorithm (development by T_A and T_B , and halting condition)

- if $\alpha \equiv_G \beta$ then $\forall\ u \in \mathrm{Dom}(T_{\alpha,\beta}),\ T_{\alpha,\beta}(u) \in \equiv_G$ (by induction on $|u| \geq 0$)
  so, every leaf of $T_{\alpha,\beta}$ is not a non-transformable pair
- conversely, if every leaf of $T_{\alpha,\beta}$ is not a non-transformable pair then every leaf is reflexive or is (or its symmetry) an internal label of $T_{\alpha,\beta}$.

  Let $R_{\alpha,\beta}$ be the set of labels of $T_{\alpha,\beta}$ having been transformed by T_A

  $S_{\alpha,\beta}$ be the set of the non-reflexive leaves of $T_{\alpha,\beta}$

  so $T_A(R_{\alpha,\beta}) \neq \emptyset$

  then $\forall\ u \in \mathrm{Dom}(T_{\alpha,\beta}),\ T_{\alpha,\beta}(u) \in \overset{*}{\underset{Q}{\longleftrightarrow}}$ with $Q = R_{\alpha,\beta} \cup S_{\alpha,\beta}$

  (by induction on $d(T_{\alpha,\beta}\backslash u) \geq 0$ with $u \in \mathrm{Dom}(T_{\alpha,\beta})$)

  moreover, $\forall\ \Phi \in S_{\alpha,\beta} - (R_{\alpha,\beta} \cup (R_{\alpha,\beta})^{-1}),\ \exists\ S \subset R_{\alpha,\beta} \cup S_{\alpha,\beta}$ such that

  $$\Phi \in \overset{*}{\underset{S}{\longleftrightarrow}} \text{ et } \forall\ \Psi \in S,\ \mu_G(\Psi) < \mu_G(\Phi) \qquad \text{(I)}$$

  in fact, for $\Phi \in S_{\alpha,\beta} - (R_{\alpha,\beta} \cup (R_{\alpha,\beta})^{-1})$, there exists one and only one internal node u of $T_{\alpha,\beta}$ such that $T_{\alpha,\beta}(u) \in \{\Phi,\Phi^{-1}\}$

  let $V = \{v \in \mathrm{Dom}(T_{\alpha,\beta}\backslash u) - \{\varepsilon\} / T_{\alpha,\beta}(uv) \in R_{\alpha,\beta} \cup S_{\alpha,\beta} \cup (S_{\alpha,\beta})^{-1}$ and

  $$\forall\ w,\ u <_{FG} w <_{FG} uv,\ T_{\alpha,\beta}(w) \notin R_{\alpha,\beta} \cup S_{\alpha,\beta} \cup (S_{\alpha,\beta})^{-1}\}$$

  then $S = \{\ T_{\alpha,\beta}(uv) / v \in V\} \subset R_{\alpha,\beta} \cup S_{\alpha,\beta} \cup (S_{\alpha,\beta})^{-1}$ and $\Phi \in \overset{*}{\underset{S}{\longleftrightarrow}}$

  in addition, the pairs of S are obtained only by T_B from $T_{\alpha,\beta}(u)$

  thus, by decreasing with μ_G of T_B, we have

  $$\forall\ \Psi \in S,\ \mu_G(\Psi) < \mu_G(\Phi) \text{ hence (I)}$$

  $T_{\alpha,\beta}$ being finite, $S_{\alpha,\beta}$ is finite.

  From (I), we deduce that $\forall\ u \in \mathrm{Dom}(T_{\alpha,\beta})\ ,\ T_{\alpha,\beta}(u) \in \overset{*}{\underset{R}{\longleftrightarrow}}$

In particular, $(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}}$ and $T_A(R_{\alpha,\beta}) \subset \overset{*}{\underset{R}{\longleftrightarrow}}$

but $T_A(R_{\alpha,\beta}) \neq \emptyset$ so $R_{\alpha,\beta}$ is self-proving, hence $\alpha \equiv_G \beta$

this completes the proof of the proposition 3.4 □

Now we define the transformation T_B

Definition 3.2 :

For any $A, B \in N$ and $\alpha,\beta \in N^+$, we define $T_B(A\alpha,B\beta)$ to be

$(T_B(B\beta,A\alpha))^{-1}$ if $A <_N B$

$\emptyset$ if $B \leq_N A$, $R_G(A,Val(B)) = \emptyset$

or $\{\gamma\}=R_G(A,Val(B))$ and $\tau_G(\gamma)\neq\tau_G(A)-\tau_G(B)$

$\{(A,B\gamma),(\gamma\alpha,\beta)\}$ if $B \leq_N A$, $\{\gamma\}=R_G(A,Val(B))$

and $\tau_G(\gamma)=\tau_G(A)-\tau_G(B)$

Thus, for any $\alpha,\beta \in N.N^+$, we have

$\alpha \equiv_G \beta \Rightarrow \emptyset \neq T_B(\alpha,\beta) \subset \equiv_G$ ($\equiv_G$ is simplifiable for the concatenation)

$(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}}$ if $R = T_B(\alpha,\beta) \neq \emptyset$

$\forall (\alpha',\beta') \in T_B(\alpha,\beta), \mu_G(\alpha',\beta') < \mu_G(\alpha,\beta)$ with $\mu_G(\lambda,\mu) = \max(\tau_G(\lambda),\tau_G(\mu))$ for any $\lambda,\mu \in N^*$

Remarks :

- if $\alpha(1) = \beta(1)$ then $T_B(\alpha,\beta) = \{(\alpha(1),\alpha(1)),(\alpha\backslash 2,\beta\backslash 2)\}$ which amounts to delete the first common letter,
- the decreasing property of T_B according to μ_G implies that we must split only couples of words in length at least equal to two

From the proposition 3.4, we only have to consider the problem of termination. We have to define an order and conditions on the choice of T_A and T_B to transform the pairs in order that the algorithm always terminates.

It suffices for T_B to be prior to T_A : we cut one pair if possible or transform it by T_A. In this case, for any (α,β), $T_{\alpha,\beta}$ is finite because, for $\alpha \equiv_G \beta$ and the set $R_{\alpha,\beta}$ of pairs having been transformed by T_A, $R_{\alpha,\beta}$ is included in $\{(\lambda,\mu) \in N\times N^+ \cup N^+\times N / \tau_G(\lambda) = \tau_G(\mu)\}$ which is finite. No condition is necessary on the order of development of the tree. From the proposition 3.4, we obtain an equivalence algorithm for $\equiv_G$ and furthermore $R = \equiv_G \cap (N\times N^+)$ is a finite (the words of an equivalent pair are equal in valuation) generating system of $\equiv_G$ on N^* : $\equiv_G = \overset{*}{\underset{R}{\longleftrightarrow}}$. In fact, for any $(\alpha,\beta) \in \equiv_G$, there exists, from the proposition 3.4, $R_{\alpha,\beta} \subset R$ / $(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}}$ and so $\equiv_G \subset \overset{*}{\underset{R}{\longleftrightarrow}}$. Conversely, $R \subset \equiv_G$

implies $\xleftrightarrow[R]{*} \subset \equiv_G$ hence the equality.

Now, let us define a minimal (for the inclusion) generating systems of $\equiv_G$ on N^* and deduce an improvement (in complexity) of the equivalence algorithm for $\equiv_G$.

We have just seen that $\equiv_G \cap (N \times N^+)$ is a generating system of $\equiv_G$. To obtain a generating system of $\equiv_G$ minimal for the inclusion, it suffices to extract from $\equiv_G \cap (N \times N^+)$ a functional system R maximal for the inclusion, but such that R does not possess symmetrical pair (then reflexive) or pair obtained by transitive closure of the others.

Theorem 2 :

Let R be a maximal subset of $\equiv_G \cap (N \times N^+)$ such that

a) $\forall (A,\alpha) \in R,\ \alpha(1) <_N A$

b) R is functional : if (A,α) , $(A,\alpha') \in R$ then $\alpha = \alpha'$

Then $\#R \leq \|G\|-1$

R is a canonical system

R generates $\equiv_G$ over N^*, that is $\equiv_G = \xleftrightarrow[R]{*}$

Proof :

i) R being included in $N \times N^+$ and being functional, $\#R \leq \#N = \|G\|$
but $\forall (A,\alpha) \in R,\ \alpha(1) <_N A$ so we have $\#R \leq \|G\|-1$

ii) Let us show that R is a canonical system
let us note that R is directed by increasing length ! However,

$$\forall (A,\alpha) \in R, \forall i \in [|\alpha|],\ \alpha(i) <_N A \quad (1)$$

in fact, by definition $\alpha(1) <_N A$
moreover, $\forall i \in [|\alpha|-1],\ \tau_G(\alpha(i+1)) \leq \tau_G(\alpha\backslash 2) < \tau_G(\alpha) = \tau_G(A)$
hence (1) is valid and R is noetherian
but R is functional, so it is confluent. Thus, R is a canonical system

iii) Let us show that R is a generating system of $\equiv_G$
by definition, $R \subset \equiv_G$ so $\xleftrightarrow[R]{*} \subset \equiv_G$
conversely, let $(\alpha,\beta) \in \equiv_G$ and prove that $(\alpha,\beta) \in \xleftrightarrow[R]{*}$
$R \subset \equiv_G$ and R is canonical, therefore $(\alpha\downarrow R, \beta\downarrow R) \in \equiv_G$
suppose that $\alpha\downarrow R \neq \beta\downarrow R$.
but $\tau_G(\alpha\downarrow R) = \tau_G(\beta\downarrow R)$ then $\alpha\downarrow R$ and $\beta\downarrow R$ are distinguishable, that is

$$\exists i \in [\min(|\alpha\downarrow R|,|\beta\downarrow R|)] / (\alpha\downarrow R)(i) \neq (\beta\downarrow R)(i) \text{ and } (\alpha\downarrow R)|(i-1) = (\beta\downarrow R)|(i-1)$$

G being reduced, we obtain $(\alpha\downarrow R)\backslash i \equiv_G (\beta\downarrow R)\backslash i$ with $(\alpha\downarrow R)(i) \neq (\beta\downarrow R)(i)$
by a symmetrical argument, we can suppose $(\beta\downarrow R)(i) <_N (\alpha\downarrow R)(i)$
by definition and validity of the transformation T_B, we get

$\exists \gamma \in N^* / ((\alpha\downarrow R)(i),(\beta\downarrow R)(i).\gamma) \in \equiv_G$

hence a contradiction by maximality of R

so $\alpha\downarrow R = \beta\downarrow R$ therefore $(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}}$

thus R is a generating system of $\equiv_G$ and the theorem 2 is proved □

The importance of such a result : $\equiv_G$ is decidable because it is finitely generated. In addition, $\equiv_G$ has canonical generating systems which have less elements than the total number of non-terminal letters of the grammar !

We will check that the relations of theorem 2 are generating systems of $\equiv_G$ which are minimals for the inclusion. From the proof of the theorem, we obtain :

Corollary :

If R is a functional subset of $N \times N^+$ such that

$\forall (A,\alpha) \in R, \alpha(1) <_N A$ and $\tau_G(A) = \tau_G(\alpha)$

Then $\#R \leq \|G\|-1$ and R is canonical

The purpose of calculating the normal form of a word with such a relation R is to have no more letters of Dom(R) : $\forall \alpha \in N^*, \alpha\downarrow R \in (N\text{-}Dom(R))^*$. The more elements R possesses (at worst $\|G\|-1$), the fewer letters for $\alpha\downarrow R$ will be left possible. Also, to improve the previous equivalence algorithm, we build the equivalence tree by reducing any pair (λ,μ) before transforming it, that is by calculating $(\lambda\downarrow R,\mu\downarrow R)$ for R the set of pairs (directed as in the corollary) transformed by T_A in the current tree. This permits to minimize the total number of pairs transformed by T_A (hence the priority of T_B on T_A) but also to limit the halting condition of one pair to the reflexives pairs (we do not test any longer if one pair has been already transformed). However, for such a modification to be valid, it is necessary for any pair transformed by T_A to have words of the same valuation. If the words of one pair have different valuations then the pair is not equivalent and, by validity of the transformations, the root pair too. As a result, we block the building of the tree if the words of one pair to be transformed have different valuation.

We can now formally define this equivalence algorithm. We will valid this algorithm and in particular, show for every $(\alpha,\beta) \in \equiv_G$ that the set $R_{\alpha,\beta}$ of labels (directed as in the corollary) of $T_{\alpha,\beta}$ transformed by T_A, is self-proving. We will deduce that $\{(A,\alpha\downarrow R)/ (A,\alpha) \in R\}$ for $R = \cup R_{\alpha,\beta}$ and $(\alpha,\beta) \in \equiv_G$ is a (reduced) canonical system generating $\equiv_G$ and minimal for the inclusion. We will evaluate the complexity of the algorithm. Of course, we do not show that this

complexity is optimal but from this algorithm, we establish a bound for Divg depending on G, bound which is minimal.

4. Equivalence algorithm for the simple grammars

We use the previous notations. We define an algorithm to decide that $(\alpha,\beta) \in \equiv_G$, for $\alpha,\beta \in N^*$, by building in lexicographic order (we could choose the parallel order) a tree $Tree_G(\alpha,\beta)$ in $N^* \times N^*$. The answer of decision is indicated by $Resp_G(\alpha,\beta) \in$ {"yes","no"}.

$(Tree_G(\alpha,\beta), Resp_G(\alpha,\beta)) = Const(\{(\varepsilon,(\alpha,\beta))\},\varepsilon,\emptyset)$ where Const is an application which for T , u , R [indicating respectively the current tree, node of tree and reduction system] associates Const(T,u,R) defined as follows :

Remarks : the assignment is represented by the symbol $\leftarrow$
the procedure Stop aborts the execution

BEGIN

1. If the words of the current label differs in valuation then we stop

$(\lambda,\mu) \leftarrow T(u)$
If $\tau_G(\lambda) \neq \tau_G(\mu)$ **Then** Stop(T,"no") **Endif**

2. We compute the pair of the normal forms of the label according to the current reduction system. Then, we cancel the longest identical left factor of the words. At last, the obtained pair is added to the tree if it is different from the label (which is more comprehensible).

If $\lambda \neq \mu$ **Then** $(\lambda',\mu') \leftarrow (\lambda\downarrow R,\mu\downarrow R)$ **Else** $(\lambda',\mu') \leftarrow (\lambda,\mu)$ **Endif**
If $\lambda' \neq \mu'$ **Then** $(\lambda',\mu') \leftarrow (\lambda'',\mu'')$ for $\lambda' = \gamma.\lambda''$, $\mu' = \gamma.\mu''$, $\lambda''(1) \neq \mu''(1)$ **Endif**
If $(\lambda',\mu') \neq (\lambda,\mu)$ **Then**
    $Dom(T) \leftarrow Dom(T) \cup \{u.1\}$; $T(u.1) \leftarrow (\lambda',\mu')$
    $u \leftarrow u.1$; $(\lambda,\mu) \leftarrow (\lambda',\mu')$
Endif

3. If the current label is reflexive then its node is a leaf and we go to the following node (by lexicographic order) if there is one.

If $\lambda = \mu$ **Then**
    $\mathcal{A} \leftarrow \{v \in Dom(T) / u <_{lex} v\}$
    If $\mathcal{A} = \emptyset$ ***Then*** Stop(T,"yes") ***Else*** $Const(T,\min_{\leq lex}\mathcal{A},R)$ ***Endif***
Endif

4. If words of the current label are not letters then we apply a T_B transformation or if not a T_A transformation. If the transformation fails then we stop or if not we continue with the daughter pair the more on the left.

If $\min(|\lambda|,|\mu|) > 1$ **Then** $\mathcal{A} \leftarrow T_B(\lambda,\mu)$ **Else**
$\mathcal{A} \leftarrow T_A(\lambda,\mu)$
If $\mu(1) <_N \lambda(1)$ ***Then*** $(\lambda',\mu') \leftarrow (\lambda,\mu)$ ***Else*** $(\lambda',\mu') \leftarrow (\mu,\lambda)$ ***Endif***
$R \leftarrow \{(\gamma\downarrow\{(\lambda',\mu')\},\delta\downarrow\{(\lambda',\mu')\}) / (\gamma,\delta) \in R\} \cup \{(\lambda',\mu')\}$
Endif
If $\mathcal{A} = \emptyset$ **Then** Stop(T,"no") **Else**
$Dom(T) \leftarrow Dom(T) \cup \{u.i / i \in [\#\mathcal{A}]\}$
$\{T(u.i) / i \in [\#\mathcal{A}]\} \leftarrow \mathcal{A}$ * of free disposition but such as in the case of a T_B , T(u.1) is the pair $(A,B\gamma)$ or $(A\gamma,B)$ with $(A,B) = (\lambda(1),\mu(1))$ *
Const(T,u.1,R)
Endif

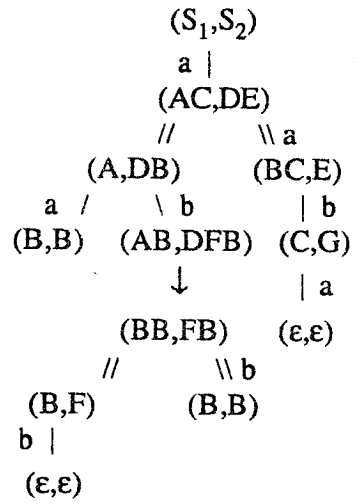
END

We are going to show that this algorithm is always with a finite termination and that for all $\alpha,\beta \in N^*$, $\alpha \equiv_G \beta$ iff $Resp_G(\alpha,\beta)$ = "yes" (and thus $\alpha \not\equiv_G \beta \Leftrightarrow Resp_G(\alpha,\beta)$ = "no"). Let's give a few examples. We represent T_A, T_B and reduction operations (computation of the normal forms and cancelation of the longest identical left factor) respectively by a bar, a double bar and an arrow. We denote by $R_G(\alpha,\beta)$ the final reduced system obtained from the root pair (α,β).

Example 1 : we use an example of Korenjak and Hopcroft; let G be the following grammar :

$$G \quad \begin{array}{lcl} S_1 \to aAC \\ A \to aB + bAB \\ B \to b \\ C \to a \end{array} \quad \cup \quad \begin{array}{lcl} S_2 \to aDE \\ D \to a + bDF \\ E \to bG \\ F \to b \\ G \to a \end{array}$$

We define $<_N$ increasing with τ_G and for the letters of the same valuation in alphabetic order : B, C, D, F, G, A, E, S_1,S_2.
The algorithm applied to (S_1,S_2) generates the following tree :



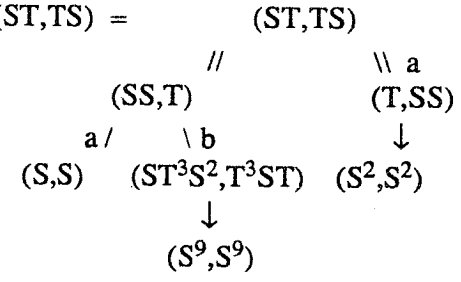
so $Resp_G(S_1,S_2)$ = "yes" and $R_G(S_1,S_2) = \{(S_1,S_2)\ ,\ (A,DB)\ ,\ (F,B)\ ,\ (E,BC)\ ,\ (G,C)\}$

Example 2 : let the following simple and reduced grammar

G S → a + bSTTTS then $\tau_G(S) = 1$

T → aS + bTTTST $\tau_G(T) = 2$

$Tree_G(ST,TS)$ =



hence $Rep_G(ST,TS)$ = "yes" and $R_G(ST,TS) = \{(T,SS)\}$

It is interesting to compare this tree with the one we could obtain with the K-H algorithm (or the one of [Wo 73]). When examples are presented to illustrate the functionning of a branching algorithm, these examples, like the previous ones, deal only with the case when the root pair is equivalent. In that case, the difficulty is to obtain a finite tree. In the other case, the difficulty is to obtain the failing in the tree, that is to say when we come to a testable inequivalent pair (non-transformable pair or pair with words of different valuation).

Example 3 :

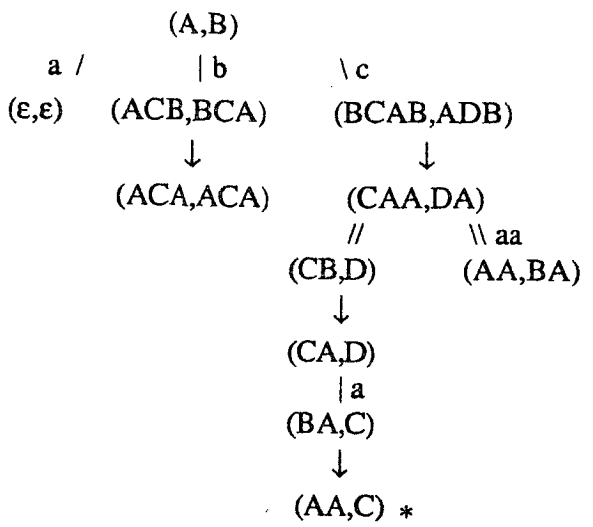
G A → a + bACB + cBCAB

B → a + bBCA + cADB

C → aB

D → aC

$Tree_G(A,B)$ =



hence $Resp_G(A,B)$ = "no" and $L(G,A) \neq L(G,B)$

5. Validity and complexity of the algorithm

We must prove that the algorithm is well defined, with a finite termination, and must end with a "yes" only for the equivalent pairs.

Proposition 5.1 :

Let $\alpha,\beta \in N^*$. $Tree_G(\alpha,\beta)$ exists and is a finite tree in $N^* \times N^*$

Proof : Consider the eventually infinite sequence $(T_i,u_i,R_i)_{i\geq 0}$ of the successive calling parameters of the recursive procedure Const with $T_0 = \{(\varepsilon,(\alpha,\beta))\}$, $u_0 = \varepsilon$ and $R_0 = \emptyset$

Let $\mathcal{R} = \{R \in P_f(N \times N^+) / R$ is functionnal and $\forall (A,\alpha) \in R$, $\alpha(1) <_N A$ and $\tau_G(\alpha) = \tau_G(A)\}$

From the corollary of the theorem 2, every relation of $\mathcal{R}$ is canonical. For $R \in \mathcal{R}$ and $(\lambda,\mu) \in N \times N^+ / (\tau_G(\lambda) = \tau_G(\mu), \mu(1) <_N \lambda(1), \lambda \downarrow R = \lambda$ and $\mu \downarrow R = \mu)$ then

$\{(\gamma\downarrow\{(\lambda,\mu)\},\delta\downarrow\{(\lambda,\mu)\}) / (\gamma,\delta) \in R\} \cup \{(\lambda,\mu)\}$ exists and belongs to $\mathcal{R}$

so, by induction on $i \in \mathbb{N}$, if R_i exists then $R_i \in \mathcal{R}$

thus, after some checking, $Tree_G(\alpha,\beta)$ exists

Let us show that $Tree_G(\alpha,\beta)$ is finite i.e. the sequence $(T_i,u_i,R_i)_{i\geq 0}$ is finite

assume that the sequence $(T_i,u_i,R_i)_{i\geq 0}$ is infinite

but $\forall (\lambda',\mu') \in T_B(\lambda,\mu)$, $\mu_G(\lambda',\mu') < \mu_G(\lambda,\mu)$ with $\mu_G(\gamma,\delta) = \max(\tau_G(\gamma),\tau_G(\delta))$ then

$\forall i \in \mathbb{N}, \exists j > i, \exists u \in Dom(T_j)\text{-}Dom(T_i) / T_j(u)$ has been transformed by T_A

thus, $\forall i \in \mathbb{N}, \exists j > i / \#R_i < \#R_j$

but $\forall i \in \mathbb{N}, R_i \in \mathcal{R}$ so $\#R_i \leq \|G\|!$ hence a contradiction □

Proposition 5.2 :

Let $\alpha,\beta \in N^*$

$\alpha \equiv_G \beta \Leftrightarrow Resp_G(\alpha,\beta) = \text{"yes"}$

Proof :

i) If $\alpha \equiv_G \beta$ then by induction on $|u| \geq 0$ for $u \in Dom(Tree_G(\alpha,\beta))$, $Tree_G(\alpha,\beta)(u) \in \equiv_G$
therefore $Resp_G(\alpha,\beta) = \text{"yes"}$

ii) Let $\alpha,\beta \in N^*$ / $Resp_G(\alpha,\beta) = \text{"yes"}$ and show that $\alpha \equiv_G \beta$
so, all the leaves of $Tree_G(\alpha,\beta)$ have reflexive labels
let R be the set of labels of $Tree_G(\alpha,\beta)$ having been transformed by T_A
let $(T_i,u_i,R_i)_{i\geq 0}$ be the finite sequence of the successive calling parameters of the recursive procedure Const with $T_0 = \{(\varepsilon,(\alpha,\beta))\}$, $u_0 = \varepsilon$ and $R_0 = \emptyset$
then, by induction on the index i of the sequence, $R_i \subset \overset{*}{\underset{R}{\longleftrightarrow}}$
thus, by induction on $d(Tree_G(\alpha,\beta)\backslash u) \geq 0$, for $u \in Dom(Tree_G(\alpha,\beta))$,
$Tree_G(\alpha,\beta)(u) \in \overset{*}{\underset{R}{\longleftrightarrow}}$
in particular, $(\alpha,\beta) \in \overset{*}{\underset{R}{\longleftrightarrow}}$ and $\emptyset \neq T_A(R) \subset \overset{*}{\underset{R}{\longleftrightarrow}}$ i.e. R is self-proving
so $R \subset \equiv_G$, thus $\overset{*}{\underset{R}{\longleftrightarrow}} \subset \equiv_G$, hence $\alpha \equiv_G \beta$ □

Remark : let $R_G(\alpha,\beta) = \{(\lambda,\mu) / \mu(1) <_N \lambda(1)$ and (λ,μ) or (μ,λ) is label of $Tree_G(\alpha,\beta)$ having been transformed by $T_A\}$
let $R' = \{(\lambda,\mu) \in R_G(\alpha,\beta) / (\alpha,\beta) \in \equiv_G\}$ which is canonical (but not functionnal)
then $R = \{(\lambda,\mu\downarrow R') / (\lambda,\mu) \in R'\}$ is a generating system of $\equiv_G$, canonical and minimal for the inclusion

We just have to evaluate the algorithm complexity in function of characteristics of the G simple grammar and (α,β) pair to be compared.

. The maximum labels of $Tree_G(\alpha,\beta)$ being transformed by T_A is no more than $\|G\|$ and it is similar for the pairs transformed by T_B. So, the size of the equivalence tree $\|Tree_G(\alpha,\beta)\|$ is in $O(\#\Sigma_G.\|G\|)$

. Adding one pair to the system of reduction R is in $O(\ell_G.\|G\|^2)$

. The maximal valuation of one pair of the tree $Tree_G(\alpha,\beta)$ is $\max(\tau_G(\alpha),\tau_G(\beta),e_G)$ with $e_G = \max\{\tau_G(\alpha) / \alpha \in Im(G)\} \leq \ell_G.|G|$. So, the computation for the set of the labels of the tree about the test on the equality of valuations, on the computation of normal forms and on the erasing of the common left factor with the equality test is in $O(\max(\tau_G(\alpha),\tau_G(\beta),\ell_G.|G|).\ \#\Sigma_G\ .\ \|G\|)$, which is

greater to the one of T_A and T_B transformations.

. $\#G \leq \|G\|.\#\Sigma_G$ so the determination of Val is in $O(|G|.\#\Sigma_G.\|G\| + \|G\|.\ell_G)$

. the complexity of the algorithm is in $O(\ell_G\|G\|^2 + \max(\tau_G(\alpha),\tau_G(\beta),\ell_G.|G|).\ \#\Sigma_G\ .\|G\|)$

Because ℓ_G is in $O(|G|^{\|G\|})$, we obtain finally

Theorem 3 :

The equivalence problem for the class of simple grammars is decidable and it exists a decision algorithm of polynomial complexity depending on the description length and valuation of the compared grammars, and of exponential complexity depending on the description length of the compared grammars

---

Remark : This theorem is also satisfied by the algorithm defined in paragraph 3 (no reduction and also a stop on the already transformed pairs).

From this algorithm (in §4), we have determine [Ca 85] a minimal bound of $Divg(\alpha,\beta)$ in $O(\ell_G.|G|.\|G\| + \min(\tau_G(\alpha),\tau_G(\beta))$. To compare this bound to the ones already known, we must restrict in case $|G| \leq 3$, $|\alpha| = |\beta| = 1$ and $\tau_G(\alpha) = \tau_G(\beta)$ for which we have $Divg(\alpha,\beta) \leq 2.\ell_G.\|G\|$ instead of $\ell_G.\|G\|^{(\ell+1)(\ell+3)}$ in [Ko-Ho 66], $\ell_G.\|G\|^{2(\ell+1)}$ in [Wo 73], $2.\ell_G.\|G\|^2$ in [Bu 73].

Thus, to work with a simple grammar, previously we could put it in canonical form [Co 74], [Co-Vu 76] by extracting a generating system of its equivalence and the cost of this "normalization" is polynomial according to the length of description and to the valuation of the grammar. Then, the decision of the equivalence of two words U and V is in $O(\min(\tau_G(U),\tau_G(V)))$.

This decreasing of complexity in deciding for the equivalence of simple grammars has no interest in itself, only if in doubt of the complexity of others existing branching algorithms, and to prove the efficiency of this current method. Now, we are going to apply this method to the equivalence problem for the stateless dpda with acceptance on pushdown letters.

6. Equivalence for stateless dpda family

A stateless dpda (dpda for "deterministic pushdown automaton") over the alphabet Σ is a quadruple $M = <N,\delta,\alpha,N_0>$ where

. N is a finite alphabet, disjoint of Σ and called **pushdown alphabet**

. δ is a function of $N\times(\Sigma\cup\{\varepsilon\})$ into N^*, called **transition function**, such that

Dom(δ) is finite and for any $A\in N$, if $(A,a) \in$ Dom(δ) with $a\in\Sigma$ then $(A,\varepsilon)\notin$ Dom(δ)

. $\alpha \in N^*$ is the initial pushdown word

. $N_0 \subset N$ is a subset of the pushdown alphabet, and called set of **acceptation letters**

We associate for each stateless dpda M, the grammar $G_M = \{(A,a\delta(A,a)) / (A,a) \in \text{Dom}(\delta)\}$. The language recognized by M on empty pushdown is $L(G_M,\alpha)$. Then, the family of languages over Σ recognized by the stateless dpda on empty pushdown is identical to the family of simple languages over Σ, plus the language reduced to the empty word. The language recognized by M on acceptation letters is $L(G_M,\alpha,N_0)$ with for any context-free grammar G over Σ, $U \in (\Sigma\cup\text{Dom}(G))^*$ and $M \subset$ Dom(G), $L(G,U,M) = \{u\in\Sigma^* / U \xrightarrow[G]{*} u\beta \text{ et } \beta(1)\in M\}$. The equivalence problem for the class of stateless dpda is the decidability of the equality in their languages recognized on acceptation letters. We define $\mathfrak{G}$ [resp. $\mathfrak{G}_0$] the family of context-free grammars G over Σ such that $G = G_S \cup G_\varepsilon$ with

. $\text{Dom}(G_S) \cap \text{Dom}(G_\varepsilon) = \emptyset$

. $\text{Im}(G_S) \subset \Sigma.(\text{Dom}(G))^*$ and $\forall A \in \text{Dom}(G_S)$, $\forall a \in \Sigma$, $\#R_G(A,a) \leq 1$

. $\text{Im}(G_\varepsilon) \subset (\text{Dom}(G))^*$ and $\forall A \in \text{Dom}(G_\varepsilon)$, $\#G_\varepsilon(\{A\}) = 1$

[resp. G_ε is reduced to only one pair which its right word is the empty word]

The equivalence problem for the stateless dpda is to know whether we can decide about the equality $L(G,\alpha,M) = L(G,\alpha',M)$ for any $G \in \mathfrak{G}$, $M \subset$ Dom(G) and $\alpha,\alpha' \in (\text{Dom}(G))^*$. In fact, for two stateless dpda over Σ, $M' = <N',\delta',\beta',Q'>$ and $M'' = <N'',\delta'',\beta'',Q''>$, we can assume (for a possible renaming) that $N' \cap N'' = \emptyset$ and to take $G = G_{M'} \cup G_{M''} \in \mathfrak{G}$, $M = Q' \cup Q''$, $\alpha = \beta'$ and $\alpha' = \beta''$. Conversely, for $G \in \mathfrak{G}$, $M \subset$ Dom(G) and $\alpha,\alpha' \in (\text{Dom}(G))^*$, we associate the two stateless dpda $M_1 = <\text{Dom}(G),\delta,\alpha,M>$ and $M_2 = <\text{Dom}(G),\delta,\alpha',M>$ with δ the transition

function defined by : $\forall A \in Dom(G)$, $\forall a \in \Sigma \cup \{\varepsilon\}$, $\delta(A,a) = \alpha$ if $(A,a\alpha) \in G$ and $\alpha \in (Dom(G))^*$.

We conclude that the equivalence problem for the stateless dpda is inter-reducible to the one of knowing if for any $G \in \mathfrak{G}_0$ and $\alpha,\alpha' \in (Dom(G))^*$, we can decide that $L(G,\alpha,\{E\}) = L(G,\alpha',\{E\})$ where (E,ε) is the pair of G. Effectively, we just have to show :

Proposition 6.1 :

$\forall G \in \mathfrak{G}$, $\forall M \subset Dom(G)$, $\exists G_0 \in \mathfrak{G}_0$ such that $\forall \alpha \in (Dom(G))^*$
$\exists \alpha_0 \in (Dom(G_0))^* / L(G,\alpha,M) = L(G_0,\alpha_0,\{E\})$ for $(E,\varepsilon) \in G_0$

Proof : Let $G \in \mathfrak{G}$ and $M \subset Dom(G)$

$Dom(G_S) \cap Dom(G_\varepsilon) = \emptyset$

so $G = G_S \cup G_\varepsilon$ / $Im(G_S) \subset \Sigma.(Dom(G))^*$; $Im(G_\varepsilon) \subset (Dom(G))^*$

$\forall A \in Dom(G_S)$, $\#R_G(A,a) \leq 1$; $\forall A \in Dom(G_\varepsilon)$, $\#G_\varepsilon(\{A\}) = 1$

We denote by E a new element $(E \notin \Sigma \cup Dom(G))$

We define $\alpha[\alpha'/A]$ the word obtained by replacing, in the word α, each letter A by the word α'

$$\alpha[\alpha'/A] = \begin{cases} \varepsilon \text{ si } \alpha = \varepsilon \\ \alpha(1).(\alpha\backslash 2[\alpha'/A]) \text{ if } \varepsilon \neq \alpha(1) \neq A \\ \alpha'.(\alpha\backslash 2[\alpha'/A]) \text{ if } \alpha(1) = A \end{cases}$$

The building of G_0 from G and M and the determination of α_0 from G, M and $\alpha \in (Dom(G))^*$ are realized by the following steps :

a) *to label the acceptation letters*

let $N_1 = \{A \in Dom(G) / \varepsilon \in L(G,A,M)\}$ so $M \subset N_1 \subset M \cup Dom(G_\varepsilon)$

and define $G_1 = \{(A,\beta[EB/B]_{B\in N1}) / (A,\beta) \in G\}$ and $\alpha_1 = \alpha[EB/B]_{B\in N1}$

b) *to delete the erasing letters*

let $N_2 = \{A \in Dom(G) / \varepsilon \in L(G,A)\}$

then $N_2 \subset Dom(G_\varepsilon)$ and more $N_2 = \{A \in Dom(G) / L(G,A) = \{\varepsilon\}\}$

take $G_2 = \{(A,\beta[\varepsilon/B]_{B\in N2}) / (A,\beta) \in G_1$ and $A \notin N_2\}$ and $\alpha_2 = \alpha_1[\varepsilon/B]_{B\in N2}$

c) *to delete the blocking letters*

let $N_3 = \{A \in Dom(G_2) / \forall a \in \Sigma, R_G(A,a) = \emptyset\} \subset Dom(G_\varepsilon) - N_2$

we take $G_3 = \{(A,\beta') / (A,\beta) \in G_2, A \notin N_3,$

β' the longest left factor of β such that $\beta' \in (\Sigma \cup \{\varepsilon\}).(Dom(G_2)-N_3)^*\}$

and α_3 the longest left factor of α_2 such that $\alpha_3 \in (Dom(G_2)\text{-}N_3)^*$

d) *to put in Greibach normal form*

$G_0 = \{(A,\beta) / A \in Dom(G_3), \beta = a\beta' \text{ for } a \in \Sigma \text{ and } \{\beta'\} = R_{G3}(A,a)\} \cup \{(E,\varepsilon)\}$

$\alpha_0 = \alpha_3$ and we have $L(G_0,\alpha_0,\{E\}) = L(G,\alpha,M)$ with $G_0 \in \mathfrak{G}_0$

thus the proposition 6.1 □

Remark : the building of G_0 [resp. α_0] linearly depends on the description length of G [resp. and of $|\alpha|$].

For any $G \in \mathfrak{G}_0$ and $U \in (\Sigma \cup Dom(G))^*$, we set up $\underset{\sim}{L}(G,U) = L(G,U,\{E\})$ where $(E,\varepsilon\}$ is the only erasing pair of G and we define the relation $\sim_G$ on $(\Sigma \cup Dom(G))^*$ by : $\forall\ U,V \in (\Sigma \cup Dom(G))^*$, $U \sim_G V$ iff $\underset{\sim}{L}(G,U) = \underset{\sim}{L}(G,V)$. So, the equivalence problem for the stateless dpda corresponds to the decidability of $\sim_G$, restricted to $(Dom(G))^*$, for any $G \in \mathfrak{G}_0$.

From now on, G is a grammar of $\mathfrak{G}_0$, $N = Dom(G)$ and E is the element of N such that $(E,\varepsilon) \in G$. $\sim_G$ is an equivalence but not a congruence : $\sim_G$ is closed under left concatenation but not under right concatenation. We determine in $O(\#G.|G|)$, $N_0 = \{A \in N / L(G,A) = \emptyset\}$, τ_G on $N_1 = N - N_0$, $M_0 = \{A \in N / \underset{\sim}{L}(G,A) = \emptyset\}$. To study $\sim_G$ on N^*, we need to put the non-terminals in a standard form : we project N^* in $\mathfrak{E} = \{\alpha \in N_1^*.(N_0 \cup \{\varepsilon\}) - N^*M_0 / EE \text{ is not a subword of } \alpha$ i.e. $\forall\ i \in [|\alpha|\text{-}1]$, if $\alpha(i) = E$ then $\alpha(i+1) \neq E\}$ with the operation $Proj_{\mathfrak{E}} : N^* \rightarrow \mathfrak{E}$. This operation is defined as follows : for any non-terminal α , $Proj_{\mathfrak{E}}(\alpha) = f_{\mathfrak{E}}(f_{M0}(f_{N0}(\alpha)))$, $f_{N0}(\alpha)$ is the longest prefix of α belonging to $N_1^*.(N_0 \cup \{\varepsilon\})$, $f_{M0}(\alpha)$ is the longest prefix of α belonging to $N^*\text{-}N^*M_0$, $f_{\mathfrak{E}}(\alpha)$ deletes the non-used E letters : $f_{\mathfrak{E}}(\alpha) = f_{\mathfrak{E}}(\alpha\backslash 2)$ if $\alpha(1) = \alpha(2) = E$ or if not $f_{\mathfrak{E}}(\alpha) = \alpha(1).f_{\mathfrak{E}}(\alpha\backslash 2)$. This projection does not alter the recognized languages : $\underset{\sim}{L}(G,\alpha) = \underset{\sim}{L}(G,Proj_{\mathfrak{E}}(\alpha))$, for any $\alpha \in N^*$. We will note that $\tau_G(E) = 0$ and $\underset{\sim}{L}(G,E) = \{\varepsilon\}$ so $E \in N_1\text{-}M_0$. Furthermore,

$\forall\ \alpha \in N^*,\ L(G,\alpha) \neq \emptyset \Leftrightarrow \alpha \in N_1^*$

$\forall\ \alpha \in N_1^*.(N_0 \cup \{\varepsilon\}),\ \underset{\sim}{L}(G,\alpha) \neq \emptyset \Leftrightarrow \alpha \notin M_0^*$

So, for any element α of $\mathfrak{E}$, $\underset{\sim}{L}(G,\alpha)$ is empty iff α is the empty word. We extend by union the projection $Proj_{\mathfrak{E}}$ to the set of finite non-terminals subsets.

As for the equivalence of a simple grammar, we define the basic transformation, denoted by T_E, which brings over the test of equivalence of a pair to the one of the set of pairs obtained by direct derivation, that is

Definition 6.1 :

$\forall\ \alpha,\beta \in \mathbb{E}$, $T_E(\alpha,\beta) = \{(\varepsilon,\varepsilon)\}$ if $\alpha = \beta \in \{E,\varepsilon\}$

$= \emptyset$ if for $\{\alpha',\beta'\} = \{\alpha,\beta\}$, $\alpha'(1) = E \neq \beta'(1)$

or $\exists\ a \in \Sigma$ / $Proj_{\mathbb{E}}(R_G(\alpha',a))-\{\varepsilon\} = \emptyset \neq Proj_{\mathbb{E}}(R_G(\beta',a))-\{\varepsilon\}$

$= \{(Proj_{\mathbb{E}}(\alpha'),Proj_{\mathbb{E}}(\beta'))$ / $\{\alpha'\}=R_G(\alpha,a),\{\beta'\}=R_G(\beta,a),a\in\Sigma\}$ else

The transformation T_E is valid : $\forall\ \alpha,\beta \in \mathbb{E}$, $\alpha \sim_G \beta \Leftrightarrow \emptyset \neq T_E(\alpha,\beta) \subset \sim_G$

So as to apply several times T_E, we extend T_E on $P_f(\mathbb{E}\times\mathbb{E})$ by

$\forall$ X finite $\subset \mathbb{E}\times\mathbb{E}$, $T_E(X) = \emptyset$ if $\exists\ (\alpha,\beta) \in X$ / $T_E(\alpha,\beta) = \emptyset$

$= \cup_{(\alpha,\beta)\in X} T_E(\alpha,\beta)$ otherwise

For any integer n, we denote $T_E^{\ n}$ the application composing T_E n times. We will write $T_E^{\ n}(\alpha,\beta)$ instead of $T_E^{\ n}(\{(\alpha,\beta)\})$. $\sim_G$ is semi-refutable (i.e. the inequivalence is semi-decidable) :

Proposition 6.2 :

Let $G \in \mathcal{G}_0$ and $\alpha,\beta \in (Dom(G))^*$

The following two properties are equivalents :

a) $\alpha \sim_G \beta$ i.e. $\underset{\sim}{L}(G,\alpha) = \underset{\sim}{L}(G,\beta)$

b) $\forall\ n \geq 0$, $T_E^{\ n}(Proj_{\mathbb{E}}(\alpha),Proj_{\mathbb{E}}(\beta)) \neq \emptyset$

This proposition is deduced from the following lemma :

Lemma 6.1 :

Let $\alpha,\beta \in \mathbb{E}$ and $n \in \mathbb{N}$

$T_E^{\ n}(\alpha,\beta) \neq \emptyset \Leftrightarrow \forall\ u \in \Sigma^*,\ 0\leq|u|<n,\ u \in \underset{\sim}{L}(G,\alpha) \Leftrightarrow u \in \underset{\sim}{L}(G,\beta)$

and $1\leq|u|\leq n$, $Proj_{\mathbb{E}}(R_G(\alpha,u))-\{\varepsilon\} \neq \emptyset \Leftrightarrow Proj_{\mathbb{E}}(R_G(\beta,u))-\{\varepsilon\} \neq \emptyset$

and in that case, $T_E^{\ n}(\alpha,\beta)-\{(\varepsilon,\varepsilon)\} =$

$\{(Proj_{\mathbb{E}}(\alpha'),Proj_{\mathbb{E}}(\beta'))/\ \{\alpha'\}=R_G(\alpha,u),\ \{\beta'\}=R_G(\beta,u),\ u \in \Sigma^n\} - \{(\varepsilon,\varepsilon)\}$

that will prove by induction on $n \geq 0$

Thus, for any $\alpha,\beta \in N^*$, $\alpha \not\sim_G \beta \Leftrightarrow \exists\ n \geq 1$ / $T_E^{\ n}(Proj_{\mathbb{E}}(\alpha),Proj_{\mathbb{E}}(\beta)) = \emptyset$. Then, we can define the left divergence operator Divg associated with T_E by : $\forall\ \alpha,\beta \in N^*$, $Divg(\alpha,\beta) = \infty$ if $\alpha \sim_G \beta$ or if not the smallest integer n for which $T_E^{\ n}(Proj_{\mathbb{E}}(\alpha),Proj_{\mathbb{E}}(\beta)) = \emptyset$. To decide if (α,β) belongs to $\sim_G$, we built via T_E transformation a tree from the root pair $(Proj_{\mathbb{E}}(\alpha),Proj_{\mathbb{E}}(\beta))$. We

stop transforming a pair if it is a final pair that is a reflexive pair or a pair (or its symmetry) already transformed in the tree. As for the T_A transformation, the transformation T_E is insufficient to obtain a finite tree for any non-terminals pair. Then, we want to introduce a cutting operation in a similar way to the T_B transformation. The difficulty is that $\sim_G$ is not a congruence : $\sim_G$ is closed under left concatenation ($\alpha \sim_G \beta \Rightarrow \gamma\alpha \sim_G \gamma\beta$) but not under right concatenation as it is shown by the following example : for G = {(A,aE) , (B,aEC) , (C,c) , (E,ε)}, we have $A \sim_G B$ but $AA \not\sim_G BA$. So, for $R \subset \sim_G$, it is possible that $\overset{*}{\underset{R}{\longleftrightarrow}} \not\subset \sim_G$. However, if $\alpha \sim_G \beta$ and $L(G,\alpha) = L(G,\beta)$ then $\alpha\gamma \sim_G \beta\gamma$ for any non-terminal γ. So, instead of the smallest congruence $\overset{*}{\underset{R}{\longleftrightarrow}}$ containing a binary relation R, we generates by R an other equivalence (which will not be a congruence).

We denote for $\alpha \in N^*$, $\overline{\alpha} = \alpha[\varepsilon/E] = \alpha\downarrow\{(E,\varepsilon)\}$ the word obtained from α by deleting the E letters. To G ,we associate the reduced and simple grammar $\overline{G} = \{(A,\overline{\alpha}) / (A,\alpha) \in G \text{ and } \alpha \in \Sigma.N_1^*\}$. So, $Dom(\overline{G}) = N_1\text{-}\{E\}$ and $\forall\, \alpha \in N_1^*$, $L(\overline{G},\overline{\alpha}) = L(G,\alpha)$.

For any binary relation R on N^*, we define the binary relation $\overset{}{\underset{R}{\Longrightarrow}}$ on N^* by :

Definition 6.2 :

$\forall\, \alpha,\beta \in N^*$, $\alpha \overset{}{\underset{R}{\Longrightarrow}} \beta$ iff $\exists\, \lambda,\mu \in N^*, \exists\, (\alpha_0,\beta_0) \in R$ / $\alpha = \lambda\alpha_0\mu$, $\beta = \lambda\beta_0\mu$ and if $\mu \neq \varepsilon$, we have in addition $\alpha_0,\beta_0 \in N_1^*$ and $\overline{\alpha}_0 \equiv_{\overline{G}} \overline{\beta}_0$

We denote by $\overset{}{\underset{R}{\Longleftrightarrow}}$ the symmetric closure of $\overset{}{\underset{R}{\Longrightarrow}}$ and $\overset{*}{\underset{R}{\Longleftrightarrow}}$ the reflexive and transitive closure of $\overset{}{\underset{R}{\Longleftrightarrow}}$. Like $\sim_G$, $\overset{*}{\underset{R}{\Longleftrightarrow}}$ is an equivalence but not a congruence since it is not closed under right concatenation. We say that a binary relation S on $\mathbb{E}$ is closed by T_E [resp. self-proving] if $\emptyset \neq T_E(S) \subset S$ [resp. $\emptyset \neq T_E(S) \subset \overset{*}{\underset{S}{\Longleftrightarrow}}$]. Let us prove that if $\sim_G$ is finitely generated on $\mathbb{E}$, that is it exists a finite binary relation R on $\mathbb{E}$ such that $\sim_G \cap (\mathbb{E}\times\mathbb{E}) = \overset{*}{\underset{R}{\Longleftrightarrow}} \cap (\mathbb{E}\times\mathbb{E})$; then $\sim_G$ is decidable on N^*

Theorem 4 :

If $\exists$ a finite binary relation R on $\mathbb{E}$ / $\sim_G = \overset{*}{\underset{R}{\Longleftrightarrow}}$ on $\mathbb{E}$
Then $\sim_G$ is decidable on N^*

Proof : as $\underset{\sim}{L}(G,\alpha) = \underset{\sim}{L}(G,Proj_{\mathbb{E}}(\alpha))$, it suffices to prove that $\sim_G$ is decidable on $\mathbb{E}$.

we will prove, for any binary relation R on $\mathbb{E}$ / $T_E(R) \neq \emptyset$, that

$\emptyset \neq T_E(\overset{*}{\underset{R}{\Longleftrightarrow}} \cap (\mathbb{E}\times\mathbb{E})) \subset \overset{*}{\underset{S}{\Longleftrightarrow}}$ where $S = R \cup T_E(R)$

hence $\overset{*}{\Longleftrightarrow}_R \cap (\mathbb{E}\times\mathbb{E})$ is closed by T_E iff R is self-proving

so, if $\exists$ R finite on $\mathbb{E}$ / $\sim_G \cap (\mathbb{E}\times\mathbb{E}) = \overset{*}{\Longleftrightarrow}_R \cap (\mathbb{E}\times\mathbb{E})$ then

$\forall \alpha,\beta \in \mathbb{E}, \alpha \sim_G \beta \Leftrightarrow \exists R_{\alpha,\beta}$ finite and self-proving / $(\alpha,\beta) \in R_{\alpha,\beta}$

as $\equiv_{\overline{G}}$ is decidable on $Dom(\overline{G})^*$, $\overset{*}{\Longleftrightarrow}_R$ is semi-decidable for any finite relation R and therefore it is semi-decidable that a finite relation is self-proving. By reusing the algorithm of theorem 1, we conclude that $\sim_G$ is decidable on $\mathbb{E}$ hence on N^* □

Let us show the proposition 3.3 is satisfied by $\overset{*}{\Longleftrightarrow}_R$, that is for any $(\alpha,\beta) \in \overset{*}{\Longleftrightarrow}_R$, it exists a pair of R of left divergence smaller than the one of (α,β)

Proposition 6.3 :

Let R be a nonempty binary relation on N^*

$\forall (\alpha,\beta) \in \overset{*}{\Longleftrightarrow}_R, \exists (\alpha_0,\beta_0) \in R$ / $Divg(\alpha_0,\beta_0) \leq Divg(\alpha,\beta)$

Proof : it suffices to rephrase the proof of the proposition 3.3 by using the Divg properties of the below lemma □

Lemma 6.2 :

Let $\alpha,\beta,\gamma \in N^*$

a) $1 \leq Divg(\alpha,\beta)$

b) $Divg(\alpha,\beta) + \tau_G(\gamma) \leq Divg(\gamma\alpha,\gamma\beta)$ if $Divg(\alpha,\beta) \neq 1$

c) $Divg(\alpha,\beta) \leq Divg(\alpha\gamma,\beta\gamma)$ if $\alpha,\beta \in N_1^*$ and $\overline{\alpha} \equiv_{\overline{G}} \overline{\beta}$

d) $\min(Divg(\alpha,\beta),Divg(\beta,\gamma)) \leq Divg(\alpha,\gamma)$

Proof : let us prove the b) inequality

let $\alpha,\beta,\gamma \in N^*$ / $Divg(\alpha,\beta) \neq 1$

. if $\gamma\alpha \sim_G \gamma\beta$ i.e. $Divg(\gamma\alpha,\gamma\beta) = \infty$ then the inequality is true

. if $\gamma\alpha \not\sim_G \gamma\beta$ so $\gamma \in N_1^*$. Let $n = Divg(\gamma\alpha,\gamma\beta) \geq 1$

from lemma 6.1 and by the symmetry between α and β, it suffices to consider the two cases :

. if $\exists u \in \Sigma^{n-1}$ / $u \in \underset{\sim}{L}(G,\gamma\alpha)$ and $u \notin \underset{\sim}{L}(G,\gamma\beta)$

but $\underset{\sim}{L}(G,\gamma\beta) \supset \underset{\sim}{L}(G,\gamma)$ so $u \notin \underset{\sim}{L}(G,\gamma)$

but $u \in \underset{\sim}{L}(G,\gamma\alpha)$ then $u = u'u''$ with $u' \in L(G,\gamma)$ and $u'' \in \underset{\sim}{L}(G,\alpha)$

$u'u'' \notin \underset{\sim}{L}(G,\gamma\beta)$ so $u'' \notin \underset{\sim}{L}(G,\beta)$

hence $Divg(\alpha,\beta) \leq |u''|+1 = n - |u'| \leq Divg(\gamma\alpha,\gamma\beta)-\tau_G(\gamma)$

. if $\exists u \in \Sigma^n$ / $Proj_E(R_G(\gamma\alpha,u))-\{\varepsilon\} \neq \emptyset = Proj_E(R_G(\gamma\beta,u))-\{\varepsilon\}$

• if $R_G(\gamma,u) \neq \emptyset$. Let $\{\gamma'\} = R_G(\gamma,u)$

then $Proj_E(\gamma'\alpha) \neq \varepsilon = Proj_E(\gamma'\beta)$

therefore $\gamma' \in N_1^*$ and $\gamma'\beta \in M_0^*$

thus $Proj_{\Sigma}(\alpha) \neq \varepsilon = Proj_{\Sigma}(\beta)$ so $Divg(\alpha,\beta) = 1$
which is excluded by hypothesis

- if $R_G(\gamma,u) = \emptyset$. As $Proj_{\Sigma}(R_G(\gamma\alpha,u))-\{\varepsilon\} \neq \emptyset$
  then $u = u'u''$ / $u' \in L(G,\gamma)$ and $u'' \neq \varepsilon$
  thus $R_G(\gamma\alpha,u) = R_G(\alpha,u'')$ and $R_G(\gamma\beta,u) = R_G(\beta,u'')$
  so $Proj_{\Sigma}(R_G(\alpha,u''))-\{\varepsilon\} \neq \emptyset = Proj_{\Sigma}(R_G(\beta,u''))-\{\varepsilon\}$
  hence $Divg(\alpha,\beta) \leq |u''| = |u|-|u'| \leq Divg(\gamma\alpha,\gamma\beta) - \tau_G(\gamma)$

in all cases, the b) inequality is true
In a similar way, we prove the others inequalities □

Just like the T_B transformation (proposition 3.4), we want to define a T cutting transformation, application of $\Sigma\times\Sigma$ into $P_f(\Sigma\times\Sigma)$ such that

$$\forall\ (\alpha,\beta) \in Dom(T),\quad \begin{array}{l} \alpha \sim_G \beta \Rightarrow \emptyset \neq T(\alpha,\beta) \subset \sim_G \\ (\alpha,\beta) \in \overset{*}{\underset{R}{\Longleftrightarrow}} \text{ if } R = T(\alpha,\beta) \neq \emptyset \\ \forall\ (\alpha',\beta') \in T(\alpha,\beta),\ \mu_G(\alpha',\beta') < \mu_G(\alpha,\beta) \end{array}$$

where μ_G is an application of $\Sigma\times\Sigma$ into $\mathbb{N}$ depending only on G

However, to define such an operation, it is necessary to be able to realize the simplification on the right for the concatenation. Under which conditions such an operation can be done and in the opposite case, how can it be cut ? Firstly, let us note

Lemma 6.3 :

Let $\alpha,\beta \in N^*$
If $u \in \underset{\sim}{L}(G,\alpha\beta) \cap FG(L(G,\alpha))$ with $\beta(1) \neq E$ or $u \notin L(G,\alpha)$
Then $u \in \underset{\sim}{L}(G,\alpha)$

Proof :

$u \in \underset{\sim}{L}(G,\alpha\beta)$ then $\exists\, \gamma \in N^*$ / $\{E\gamma\} = R_G(\alpha\beta,u)$
$u \in FG(L(G,\alpha))$ so let $\{\gamma'\} = R_G(\alpha,u)$
thus $\gamma'\beta = E\gamma$
but $\beta(1) \neq E$ or $u \notin L(G,\alpha)$ then $\gamma' \neq \varepsilon$
therefore $\gamma'(1) = E$ hence $u \in \underset{\sim}{L}(G,\alpha)$ □

Let us define for any $\alpha \in N^*$, $<\alpha>$ the longest suffix of α / $<\alpha>(1) \neq E$

$$\text{and } E_\alpha = E \text{ if } \alpha(1) = E$$
$$= \varepsilon \text{ if } \alpha(1) \neq E$$

So, for any $\alpha \in N^*$, $\alpha \sim_G E_\alpha<\alpha>$
We can give the principal properties of $\sim_G$ for the concatenation

Proposition 6.4 :

Let $\alpha, \beta, \gamma \in N^*$

a) $\alpha \sim_G \beta \Rightarrow \gamma\alpha \sim_G \gamma\beta$

b) $\gamma \in N_1^*$ and $\gamma\alpha \sim_G \gamma\beta \Rightarrow \langle\alpha\rangle \sim_G \langle\beta\rangle$

c) if $\alpha,\beta \in N_1^*$ and $\overline{\alpha} \equiv_{\overline{G}} \overline{\beta}$

$\alpha \sim_G \beta \Rightarrow \alpha\gamma \sim_G \beta\gamma$

$\gamma(1) \neq E$ and $\alpha\gamma \sim_G \beta\gamma \Rightarrow \alpha \sim_G \beta$

d) if $\alpha,\beta \in N_1^*$ and $\overline{\alpha} \equiv_{\overline{G}} \overline{\beta}$

$\alpha\gamma \sim_G \beta\gamma$

$\alpha\gamma' \sim_G \beta\gamma' \Rightarrow \gamma \sim_G \gamma'$

$E_\gamma = E_{\gamma'}$

Proof :

i) Let us prove the second implication of c)

Let $\gamma \in N^*$, $\alpha,\beta \in N_1^*$ / $\overline{\alpha} \equiv_{\overline{G}} \overline{\beta}$, $\alpha\gamma \sim_G \beta\gamma$ and $\gamma(1) \neq E$ and show that $\alpha \sim_G \beta$.

$\underset{\sim}{L}(G,\alpha) \subset \underset{\sim}{L}(G,\alpha\gamma) = \underset{\sim}{L}(G,\beta\gamma)$

so for $u \in \underset{\sim}{L}(G,\alpha)$, $u \in \underset{\sim}{L}(G,\beta\gamma)$

moreover $u \in FG(L(G,\alpha)) = FG(L(G,\beta))$ since $\overline{\alpha} \equiv_{\overline{G}} \overline{\beta}$

but $\gamma(1) \neq E$ then from lemma 6.3, $u \in \underset{\sim}{L}(G,\beta)$

thus $\underset{\sim}{L}(G,\alpha) \subset \underset{\sim}{L}(G,\beta)$ and by symmetry between α and β, $\alpha \sim_G \beta$

So, the second implication of c) is valid

We will prove a), b) and the first implication of c) in a similar way

ii) Let $\alpha,\beta,\delta \in N^*$ / $\alpha \sim_G \delta\alpha$, $\beta \sim_G \delta\beta$ and $\langle\delta\rangle \neq \varepsilon$ (i.e. $\delta \notin \{E\}^*$) and show that $\alpha \sim_G \beta$

$\sim_G$ being an equivalence closed under left concatenation, we have

$\forall i \in \mathbb{N}$, $\alpha \sim_G \delta^i\alpha$ and $\beta \sim_G \delta^i\beta$

Let $u \in \underset{\sim}{L}(G,\alpha) = \underset{\sim}{L}(G,\delta^{|u|+1}\alpha)$

so $u \in FG(L(G,\delta^{|u|+1}\alpha)) \Rightarrow u \in FG(L(G,\delta^{|u|+1})) - L(G,\delta^{|u|+1})$

because $\langle\delta\rangle \neq \varepsilon$ and $\overline{G}$ is in GNF

then, from lemma 6.3, $u \in \underset{\sim}{L}(G,\delta^{|u|+1}) \subset \underset{\sim}{L}(G,\delta^{|u|+1}.\beta) = \underset{\sim}{L}(G,\beta)$

thus $\underset{\sim}{L}(G,\alpha) \subset \underset{\sim}{L}(G,\beta)$ and by symmetry between α and β, $\alpha \sim_G \beta$

iii) Let us establish d)

Let $\alpha,\beta \in N_1^*$ / $\overline{\alpha} \equiv_{\overline{G}} \overline{\beta}$. Let $\gamma,\gamma' \in N^*$ / $E_\gamma = E_{\gamma'}$, $\alpha\gamma \sim_G \beta\gamma$, $\alpha\gamma' \sim_G \beta\gamma'$ and show that $\gamma \sim_G \gamma'$.

$\overline{\alpha} \equiv_{\overline{G}} \overline{\beta} \Leftrightarrow L(G,\alpha) \neq L(G,\beta)$

let $X = [L(G,\alpha)-L(G,\beta)] \cup [L(G,\beta)-L(G,\alpha)]$. Then $X \neq \emptyset$

take $u \in X$ with minimal length : $\forall v \in X, |u| \leq |v|$

by symmetry between α and β, we can assume that $u \in L(G,\alpha) - L(G,\beta)$

two cases must be consider :

. either $R_G(\beta,u) = \emptyset$

then, by definition of u and $L(G,\alpha)$ being prefix, we obtain $R_G(\beta\gamma,u) = \emptyset$

but $\alpha\gamma \sim_G \beta\gamma$ so $Proj_E(R_G(\alpha\gamma,u))-\{\varepsilon\} = \emptyset$

as $u \in L(G,\alpha)$ then for α' the element of $R_G(\alpha,u)$, we have

$R_G(\alpha\gamma,u) = \{\alpha'\gamma\}$ so $Proj_E(\alpha'\gamma) = \varepsilon$

but $\alpha' \in \{E\}^*$ so $Proj_E(\gamma) = \varepsilon$

in a same fashion, for $\alpha\gamma' \sim_G \beta\gamma'$, we obtain $Proj_E(\gamma') = \varepsilon$

hence $\underset{\sim}{L}(G,\gamma) = \underset{\sim}{L}(G,\gamma') = \emptyset$ i.e. $\gamma \sim_G \gamma'$

. or $R_G(\beta,u) \neq \emptyset$

let δ be the element of $R_G(\beta,u)$ and α' the one of $R_G(\alpha,u)$

but $<\delta> \neq \varepsilon$ and $<\alpha'> = \varepsilon$

therefore $\alpha\gamma \sim_G \beta\gamma \Rightarrow \alpha'\gamma \sim_G \delta\gamma \Rightarrow <\alpha'\gamma> \sim_G <\delta\gamma>$ i.e. $<\gamma> \sim_G <\delta>.\gamma$

so $\gamma \sim_G (E_\gamma<\delta>)\gamma$

similarly $\gamma' \sim_G (E_{\gamma'}<\delta>)\gamma' \Rightarrow \gamma \sim_G \gamma'$ from ii)

but $E_\gamma = E_{\gamma'}$ and $<\delta> \neq \varepsilon$

hence iii) and the proof of the proposition 6.4 is terminated □

From c) and d) of proposition 6.4, we are going to define two cutting transformations. The choice between these two transformations is determined by the help of the congruence $\equiv_{\overline{G}}$ which is decidable.

Proposition 6.5 :

Let $A\alpha,B\beta \in \mathfrak{E}$ / $E \neq B <_N A$ and $A\alpha \sim_G B\beta$

a) if $B \in N_1$, $\{\gamma\} = R_G(A,Val(B))$ exists, $\gamma \in N_1^*$ and $A \equiv_{\overline{G}} B\overline{\gamma}$
then $<\gamma\alpha> \sim_G <\beta>$ and $A.E_\alpha \sim_G B.E_\beta.<\gamma E_\alpha>$

b) else for $A\lambda,B\mu \in \mathfrak{E}$ / $A\lambda \sim_G B\mu$, $E_\alpha = E_\lambda$, $E_\beta = E_\mu$
we have $\alpha \sim_G \lambda$ and $\beta \sim_G \mu$

Proof :

i) assume that the condition of a) is verified

let $\{\gamma'\} = R_G(B,Val(B))$ so $\gamma' \in \{E\}^*$

$A\alpha \sim_G B\beta \Rightarrow \gamma\alpha \sim_G \gamma'\beta \Rightarrow <\gamma\alpha> \sim_G <\gamma'\beta> = <\beta>$

but $\alpha \sim_G E_\alpha<\alpha> \Rightarrow \gamma\alpha \sim_G \gamma E_\alpha<\alpha> \Rightarrow <\gamma\alpha> \sim_G <\gamma E_\alpha<\alpha>> = <\gamma E_\alpha><\alpha>$

moreover $A\alpha \sim_G B\beta \sim_G BE_\beta<\beta> \sim_G BE_\beta<\gamma\alpha>$

thus $(AE_\alpha)<\alpha> \sim_G (BE_\beta<\gamma E_\alpha>)<\alpha>$

but $\overline{AE}_\alpha = A \equiv_{\overline{G}} B\overline{\gamma} = \overline{BE_\beta<\gamma E_\alpha>}$ and $<\alpha>(1) \neq E$

from c) of proposition 6.4, $AE_\alpha \sim_G BE_\beta<\gamma E_\alpha>$

ii) assume that the condition of a) is not satisfied

. **either** $A \in N_0$. But $A\alpha, A\lambda \in \mathbb{E}$, $\alpha = \lambda = \varepsilon$ hence $\alpha \sim_G \lambda$

moreover $B\beta \sim_G A \sim_G B\mu$

- ***either*** $B \in N_0$ then $\beta = \mu = \varepsilon$ so $\beta \sim_G \mu$
- ***or*** $B \in N_1$ then from b) of proposition 6.4, $<\beta> \sim_G <\mu>$
  but $E_\beta = E_\mu$ so $\beta \sim_G \mu$

. **or** $A \in N_1$

so $B \in N_1$ ($<_N$ is directed by increasing τ_G)

thus, to prove that $\alpha \sim_G \lambda$ and $\beta \sim_G \mu$, it suffices to show that

$<\alpha> \sim_G <\lambda>$ <u>or</u> $<\beta> \sim_G <\mu>$

in fact, assume that $<\alpha> \sim_G <\lambda>$

as $E_\alpha = E_\lambda$ then $\alpha \sim_G \lambda$

since $\sim_G$ is closed under left concatenation, $A\alpha \sim_G A\lambda$

therefore $B\beta \sim_G B\mu$

since $B \in N_1$, by b) proposition 6.4, we have $<\beta> \sim_G <\mu>$

as $E_\beta = E_\mu$ then $\beta \sim_G \mu$

similarly, if $<\beta> \sim_G <\mu>$ then $\beta \sim_G \mu$ and $\alpha \sim_G \lambda$

- ***either*** $R_G(A,Val(B)) = \emptyset$
  since $A\alpha \sim_G B\beta$ and $A\lambda \sim_G B\mu$, we obtain $Proj_E(\beta) = Proj_E(\mu) = \varepsilon$
  but $B\beta, B\mu \in E$ and $B \in N_1$ then $\beta = \mu = \varepsilon$
- ***or*** $R_G(A,Val(B)) \neq \emptyset$
  let $\{\gamma\} = R_G(A,Val(B))$
  by proof i), we obtain
  $<\gamma\alpha> \sim_G <\beta>$ and $(AE_\alpha)<\alpha> \sim_G (BE_\beta<\gamma E_\alpha>)<\alpha>$
  $<\gamma\lambda> \sim_G <\mu>$ and $(AE_\alpha)<\lambda> \sim_G (BE_\beta<\gamma E_\alpha>)<\lambda>$
  . <u>either</u> $\tau_G(\gamma) = \infty$ then $<\beta> \sim_G <\gamma> \sim_G <\mu>$
  . <u>or</u> $\tau_G(\gamma) < \infty$ i.e. $\gamma \in N_1^*$ and by b) hypothesis $A \cong_{\overline{G}} B\overline{\gamma}$
  i.e. $\overline{AE}_\alpha \cong_{\overline{G}} \overline{BE}_\beta\overline{<\gamma E_\alpha>}$
  but $E_\alpha = E_\lambda$ and by d) of proposition 6.4, $<\alpha> \sim_G <\lambda>$

that completes the proof of ii) and valids the proposition □

We define the application τ_f of N^* into $\mathbb{N}$ which for every non-terminal α associates $\tau_f(\alpha) = \tau(\alpha')$ where α' is the longest prefix of α belonging to N_1^*. We extend τ_f to $N^* \times N^*$ by : $\forall\ \alpha,\beta \in N^*$, $\tau_f(\alpha,\beta) = \max(\tau_f(\alpha),\tau_f(\beta))$. From the proposition 6.5, to cut a pair $(A\alpha,B\beta) \in \mathbb{E}\times\mathbb{E}$ with $B <_N A$

- if $A \equiv_{\overline{G}} B\overline{\gamma}$ with $\{\gamma\} = R_G(A,Val(B))$ and $\gamma \in N_1^*$
  then we define $T(A\alpha,B\beta) = \{(Proj_E(<\gamma\alpha>),<\beta>)\ ,\ (AE_\alpha,BE_\beta Proj_E(<\gamma E_\alpha>))\}$
  thus $T(A\alpha,B\beta) \subset E \times E$ and $(A\alpha,B\beta) \in \overset{*}{\underset{R}{\Longleftrightarrow}}$ with $R = T(A\alpha,B\beta)$
  moreover $\tau_f(AE_\alpha,BE_\beta<\gamma E_\alpha>) = \tau(A)$ and $\tau_f(<\gamma\alpha>,<\beta>) < \tau_f(A\alpha,B\beta)$

- else
  . either we have never transformed a pair with such a form $(A\lambda,B\mu)$ with $E_\lambda = E_\alpha$ and $E_\mu = E_\beta$
    then $(A\alpha,B\beta)$ is transformed by T_E
  . or such a pair exists and then we define $T(A\alpha,B\beta) = \{(\alpha,\lambda)\ ,\ (\mu,\beta)\}$
    in that case, the transformation T is no more defined locally but relatively to the pairs already transformed in the tree. Rather than reformulating the proposition 3.4, we will notice for $R = T(A\alpha,B\beta) \cup \{(A\lambda,B\mu)\}$, that $(A\alpha,B\beta) \in \overset{*}{\underset{R}{\Longleftrightarrow}}$ and furthermore $\max(\tau_f(\alpha,\lambda),\tau_f(\mu,\beta)) < \max(\tau_f(A\alpha,B\beta),\tau_f(A\lambda,B\mu))$

Like the previous algorithm, the one we are going to define will build progressively with the tree a (canonical) reduction system which allows to reduce the pairs before transforming them. We represent by $\mathfrak{R}$ the family of possible reduction systems, that is the family of binary relations on N^* such that

$R \in \mathfrak{R}$ iff $R \subset \equiv_{\overline{G}} \cap [(E \cap N_1.\{\varepsilon,E\}) \times E]$ with

if $AE \in Dom(R)$ then $A \notin Dom(R)$ and $R(\{AE\}) = \{A\}$

if $A \in Dom(R)$ then $AE \notin Dom(R)$ and $R(\{A\}) = \{\alpha\}\ /\ E \neq \alpha(1) <_N A$

Thus, $\forall R \in \mathfrak{R}$, R is canonical and $\overset{}{\underset{R}{\longrightarrow}} = \overset{}{\underset{R}{\Longrightarrow}}$.

We build progressively a reduction system with a procedure of name Add and of parameters $R \in \mathfrak{R}$ and $(\alpha,\beta) \in \equiv_{\overline{G}}$ with $\alpha \in E \cap N_1.\{\varepsilon,E\}$, $\beta \in E$, $\alpha\downarrow R = \alpha \neq \beta\downarrow R = \beta$.

Add(R,(α,β)) is defined by the following algorithm :

BEGIN

* we direct the pair to add (noetherianity problem) *

Case depending on $(\alpha(1),\beta(1))$ belongs to

$=$: $(\lambda,\mu) \leftarrow (\alpha(1)E,\alpha(1))$

$<_N$: $(\lambda,\mu) \leftarrow (\beta,\alpha)$

$_N>$: $(\lambda,\mu) \leftarrow (\alpha,\beta)$

Endcase

$\mathcal{A} \leftarrow \{(\lambda,\mu)\}$

* case where λ is reduced to one letter and that this letter is the first letter of a word which is left member of a pair of R (confluence problem) *

If $(\lambda(1)E,\lambda(1)) \in R$ **Then**

$R \leftarrow R - \{(\lambda(1)E,\lambda(1))\}$

$A \leftarrow \mu(|\mu|)$ * last letter of μ ($\neq \lambda(1)$) *

If $A \neq E$ and $(AE,A) \notin R$ ***Then*** $\mathcal{A} \leftarrow \mathcal{A} \cup \{(AE,A)\}$ ***Endif***

Endif

$R \leftarrow \{(Proj_E(\gamma\downarrow\mathcal{A}),Proj_E(\delta\downarrow\mathcal{A})) / (\gamma,\delta) \in R\} \cup \mathcal{A}$

Return(R)

END

So, $Add(R,(\alpha,\beta)) \in \mathfrak{R}$ and is included in $\sim_G$ if $R \cup \{(\alpha,\beta)\}$ it is as well.
The simplification of a pair $(\alpha,\beta) \in E \times E$ according to $R \in \mathfrak{R}$ is done by the procedure :

$$Simpl((\alpha,\beta),R) = (\alpha,\beta) \text{ if } \alpha = \beta$$
$$= g(Proj_E(\alpha\downarrow R),Proj_E(\beta\downarrow R)) \text{ if } \alpha \neq \beta$$

with $g(\alpha,\beta) = (\alpha,\beta)$ if $\alpha = \beta$

(α',β') if $\alpha = \gamma\alpha'$, $\beta = \gamma\beta'$, $\alpha'(1) \neq E$, $\beta'(1) \neq E$ and $\alpha'(1)E_{\alpha'\backslash 2} \neq \beta'(1)E_{\beta'\backslash 2}$

So, the simplification of (α,β) consists in computing the normal forms according to a reduction system R and to delete the longest common left factor such that $Simpl((\alpha,\beta),R) \in \sim_G$ if $R \cup \{(\alpha,\beta)\} \subset \sim_G$. We can now define the equivalence algorithm of $\sim_G$. We will validate this algorithm and will evaluate its complexity. For simplicity, we have not finished off the algorithm : the set of transformed pairs by T_E of an equivalence tree when the root is equivalent, is not in any case a self-proving system.

7. Decision algorithm for $\sim_G$

We assume to have extracted a generating system of $\equiv_{\overline{G}}$. We define an algorithm, of parameters $\alpha,\beta \in N^*$, that decide if $(\alpha,\beta) \in \sim_G$ by building in lexicographic order (we could choose the parallel order) a tree in $\mathbb{E}\times\mathbb{E}$, called equivalence tree and denoted by $Tree_G(\alpha,\beta)$. The answer of decision is indicated by $Resp_G(\alpha,\beta) \in \{"yes","no"\}$.

$(Tree_G(\alpha,\beta),Resp_G(\alpha,\beta)) = Const(\{(\varepsilon,(Proj_{\mathbb{E}}(\alpha),Proj_{\mathbb{E}}(\beta)))\},\varepsilon,\emptyset)$ where Const is an application which for T, u, R [indicating respectively the current tree, node of tree and reduction system] associates Const(T,u,R) defined as follows :

BEGIN

$(\lambda,\mu) \leftarrow T(u)$

1. We simplify the current label and we add the obtained pair to the tree (more comprehensible) if it differs from the label

$(\lambda',\mu') \leftarrow Simpl((\lambda,\mu),R)$

If $(\lambda',\mu') \neq (\lambda,\mu)$ **Then**

$Dom(T) \leftarrow Dom(T) \cup \{u.1\}$; $T(u.1) \leftarrow (\lambda',\mu')$

$u \leftarrow u.1$; $(\lambda,\mu) \leftarrow (\lambda',\mu')$

Endif

2. We test the pair is of a trivial inequivalence

If $\lambda(1) \neq \mu(1)$ and $\{\lambda(1),\mu(1)\} \cap \{\varepsilon,E\} \neq \emptyset$ **Then** Stop(T,"no") **Endif**

3. If the current label is reflexive or has already been transformed (or its symmetry) in the tree so then its node is a leaf and, if there is one, we go to the following node (by lexicographic order)

If $(\lambda = \mu)$ or $(\exists\, v \in Dom(T) / v <_{lex} u$ and $(\lambda,\mu) \in \{T(v),(T(v))^{-1}\})$

Then $\mathcal{A} \leftarrow \{v \in Dom(T) / u <_{lex} v\}$

If $\mathcal{A} = \emptyset$ ***Then*** Stop(T,"yes") ***Else*** $Const(T, \min_{\leq lex} \mathcal{A}, R)$ ***Endif***

Endif

4. Transformation of the current pair

Let S be the set of labels of T having been transformed by T_E

If $\exists\ (\lambda',\mu') \in S \cup S^{-1}\ /\ \lambda'(1)E_{\lambda'\backslash 2} = \lambda(1)E_{\lambda\backslash 2}$ and $\mu'(1)E_{\mu'\backslash 2} = \mu(1)E_{\mu\backslash 2}$

Then * pair to be cutted : this pair being simplified, it cannot belong to $R \cup R^{-1}$ *

$Dom(T) \leftarrow Dom(T) \cup \{u.1,u.2\}$

$T(u.1) \leftarrow (<\lambda\backslash 2>,<\lambda'\backslash 2>)$; $T(u.2) \leftarrow (<\mu'\backslash 2>,<\mu\backslash 2>)$

Else

If $(A',B'\gamma) \in \equiv_{\overline{G}}$ for $\{A',B'\} = \{\lambda(1),\mu(1)\}$, $B' \leq_N A'$ and $\{\gamma\} = R_G(A',Val(B'))$

Then * the pair must be split *

If $A' = \lambda(1)$ Then $s = 1$ Else $s = -1$ Endif

$(\alpha',\beta') \leftarrow (\lambda\backslash 2,\mu\backslash 2)^s$

* there is no need to split the current pair if the following condition is verified *

If $<\alpha'> = \varepsilon$, $\lambda,\mu \in N_1^+$, $(\overline{\lambda},\overline{\mu}) \in \equiv_{\overline{G}}$ Then $\gamma \leftarrow \beta'$ Endif

$\Phi \leftarrow (A'E_{\alpha'},B'E_{\beta'}.Proj_E(<\gamma E_{\alpha'}>))$

$R \leftarrow Add(R,\Phi)$

If $\Phi^s \neq (\lambda,\mu)$ Then

$Dom(T) \leftarrow Dom(T) \cup \{u.1,u.2\}$

$T(u.1) \leftarrow \Phi^s$; $T(u.2) \leftarrow (Proj_E(<\gamma\alpha'>),<\beta'>)^s$

$u \leftarrow u.1$

Endif

Endif

* pair to be transformed by T_E *

$\mathcal{A} \leftarrow T_E(T(u))$

If $\mathcal{A} = \emptyset$ ***Then*** Stop(T,"no") ***Endif***

$Dom(T) \leftarrow Dom(T) \cup \{u.i\ /\ i \in [\#\mathcal{A}]\ \}$

$\{T(u.i)\ /\ i \in [\#\mathcal{A}]\} \leftarrow \mathcal{A}$

Endif

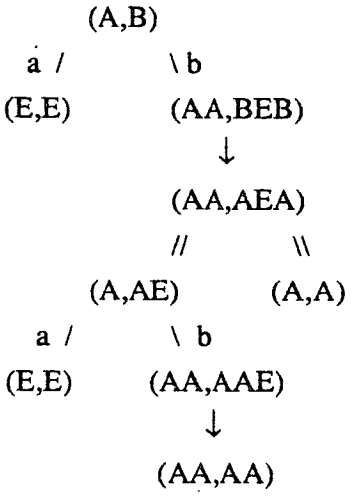
Const(T,u.1,R)

END

We are going to prove that this algorithm always ends up and for any $\alpha,\beta \in N^*$, $\alpha \sim_G \beta$ iff $Resp_G(\alpha,\beta)$ = "yes" (so $\alpha \not\sim_G \beta$ iff $Resp_G(\alpha,\beta)$ = "no"). Before hand, let us give a few examples. We represent T_E operation by a bar, cutting by a double bar and Simpl by an arrow. The pairs building up the reduction system (subset of the pairs having been transformed by T_E) are framed.

Example 1 : let G $A \rightarrow aE + bAA$ (and $E \rightarrow \varepsilon$)
$B \rightarrow aE + bBEB$

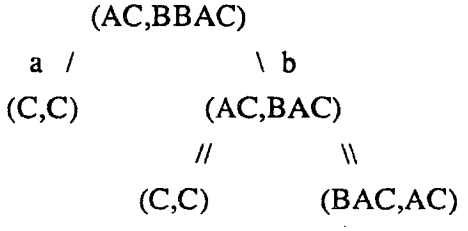
we have $N_0 = M_0 = \emptyset$ and $\tau_G(A) = \tau_G(B) = 1$. We take $A <_N B$.
$\overline{G}$ has for generating system {(B,A)}



so $Resp_G(A,B)$ = "yes" then $A \sim_G B$

Example 2 : $A \rightarrow a + bA$
let G $B \rightarrow aC + b$
$C \rightarrow aEC + bEC$

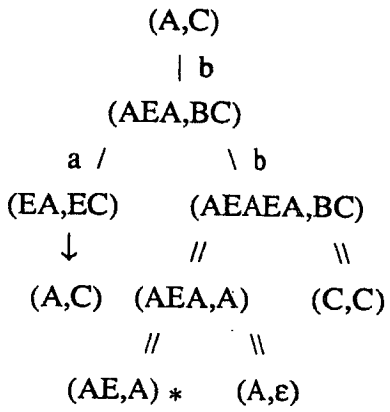
We have $N_0 = \{C\}$, $N_1 = \{A,B\}$, $M_0 = \{A\}$
$\overline{G} = \{(A,a), (A,bA), (B,b)\}$ has for generating system $\emptyset$



so AC $\sim_G$ BBAC. We will notice that (AC,BAC) $\notin \overset{*}{\underset{R}{\Longleftrightarrow}}$ with R = {(AC,BBAC)}

Example 3 : A → a + bAEA
let G B → aE + bB
C → bBC

$N_1 = \{A,B\}$, $N_0 = \{C\}$, $M_0 = \emptyset$
$\tau_G(A) = \tau_G(B) = 1$. We take $A <_N B$



so $Resp_G(A,C)$ = "no" then A $\not\sim_G$ C

8. Validity and complexity of the algorithm

We prove that the algorithm is well defined with a finite termination and that its output is "yes" if, and only if, the root pair is equivalent.

Proposition 8.1 :

Let $\alpha,\beta \in N^*$
$Tree_G(\alpha,\beta)$ exists and is a finite tree in $\mathfrak{E}\times\mathfrak{E}$

Proof :

Let us consider the possibly infinite sequence $(T_i,u_i,R_i)_{i\geq 0}$ of sucessive calling parameters of the recursive procedure Const with $T_0 = \{(\varepsilon,(Proj_{\mathfrak{E}}(\alpha),Proj_{\mathfrak{E}}(\beta)))\}$, $u_0 = \varepsilon$ and $R_0 = \emptyset$. Then $\forall\, i \geq 0$ / (T_i,u_i,R_i) exists, we have $R_i \in \mathfrak{R}$ so the procedures Add and Simpl are well defined.

Let us show that the sequence $(T_i,u_i,R_i)_{i\geq 0}$ is finite

Because $Tree_G(\alpha,\beta)$ is built without backtrack, we obtain :

(1) $\forall\, i \geq 0$, if $(T_{i+1},u_{i+1},R_{i+1})$ exists then
$Dom(T_i) \subset Dom(T_{i+1})$
$\forall\, u \in Dom(T_i),\ T_{i+1}(u) = T_i(u)$
$\#R_i \leq \#R_{i+1}$

For any integer i such that (T_i,u_i,R_i) must exist, let S_i be the set of pairs of T_i having been transformed by T_E. For any (λ,μ) , $(\lambda',\mu') \in S_i \cup (S_i)^{-1}$, if $\lambda(1)E_{\lambda\backslash 2} = \lambda'(1)E_{\lambda\backslash 2}$ and $\mu(1)E_{\mu\backslash 2} = \mu'(1)E_{\mu'\backslash 2}$ then we have $(\lambda,\mu) = (\lambda',\mu')$. We deduce that

(2) $\forall\, i \geq 0$ / (T_i,u_i,R_i) exists, $\#S_i \leq 2(\#G)^2$

Then, let us note $e_G = \max\{\tau_f(\alpha) / \alpha \in Im(G)\}$ the maximal finite valuation of the right members of the production rules of G. Then, by induction on integer i such that (T_i,u_i,R_i) must exist

(3) $\forall\, u \in Dom(T_i),\ \tau_f(T_i(u)) \leq (\#S_i)e_G + \tau_f(\alpha,\beta)$

So, from (2) and (3), we obtain : $\forall\, i \geq 0$ / (T_i,u_i,R_i) exists, $\forall\, u \in Dom(T_i),\ \tau_f(T_i(u)) \leq b_G$ where $b_G = 2(\#G)^2e_G + \tau_f(\alpha,\beta)$. Because $\{(\lambda,\mu) \in \mathfrak{E}\times\mathfrak{E} / \tau_f(\lambda,\mu) \leq b_G\}$ is finite and that from the stopping condition of a pair transformation, two internal pairs of T_i , $i \geq 0$, cannot be similar, then the sequence $(T_i,u_i,R_i)_{i\geq 0}$ is finite; which concludes the proof of proposition 8.1 □

Then, the algorithm is with a finite termination. For this algorithm to decide on $\sim_G$, iwe are left to prove that

Proposition 8.2 :

Let $\alpha,\beta \in N^*$
$\alpha \sim_G \beta \Leftrightarrow Resp_G(\alpha,\beta) =$ "yes"

Proof :

i) If $\alpha \sim_G \beta$ then, by propositions 6.4 and 6.5 and by validity of T_E, by induction on $|u| \geq 0$ for $u \in Dom(Tree_G(\alpha,\beta))$, $(Tree_G(\alpha,\beta))(u) \in \sim_G$. Thus, $Resp_G(\alpha,\beta) =$ "yes"

ii) Let $\alpha,\beta \in N^* / Resp_G(\alpha,\beta) =$ "yes" and show that $\alpha \sim_G \beta$.

Let $S_{\alpha,\beta}$ be the set of labels of $Tree_G(\alpha,\beta)$ having been transformed by T_E. From example 2, $S_{\alpha,\beta}$ cannot be self-proving. Whereas, let us show that

(I) if $\exists u \in Dom(Tree_G(\alpha,\beta)) / Tree_G(\alpha,\beta)(u) \notin \sim_G$
then $\exists v \in Dom(Tree_G(\alpha,\beta)) / Divg(Tree_G(\alpha,\beta)(v)) < Divg(Tree_G(\alpha,\beta)(u))$

assume that it exists $u \in Dom(Tree_G(\alpha,\beta)) / Tree_G(\alpha,\beta)(u) \notin \sim_G$
and let $a = Divg(Tree_G(\alpha,\beta)(u))$.
let $\bar{u} = \min_{\leq lex} \{v \in Dom(Tree_G(\alpha,\beta)) / Divg(Tree_G(\alpha,\beta)(v)) = a\}$
so, $\bar{u}$ is not a leaf of the tree and $1 \leq a \leq \infty$
let $(\lambda,\mu) = Simpl(Tree_G(\alpha,\beta)(\bar{u}),R_n)$ where $n = \max \{i / u_i \leq_{lex} \bar{u}\}$ with $(T_i,u_i,R_i)_{i\geq 0}$ sequence of successive calling parameters of Const with $T_0 = \{(\varepsilon,(\alpha,\beta))\}$, $u_0 = \varepsilon$, $R_0 = \emptyset$.
then (λ,μ) is label of the tree and we have $Tree_G(\alpha,\beta)(\bar{u}) \in \overset{*}{\underset{S}{\Longleftrightarrow}}$ where $S = R_n \cup \{(\lambda,\mu)\}$
since $R_n \subset \equiv_{\bar{G}}$ and by proposition 6.3 and definition of $\bar{u}$, we obtain
$Divg(\lambda,\mu) \leq Divg(Tree_G(\alpha,\beta)(\bar{u})) = a$

- if $Divg(\lambda,\mu) < a$ then (I) is true
- if $Divg(\lambda,\mu) = a$. Since $Resp_G(\alpha,\beta) =$ "yes" and by definition of $\bar{u}$, (λ,μ) is not a leaf of the tree and consequently has been transformed

thus $\exists (\lambda',\mu') \in S_{\alpha,\beta} \cup (S_{\alpha,\beta})^{-1} / \lambda'(1)E_{\lambda'\backslash 2} = \lambda(1)E_{\lambda\backslash 2}$ and $\mu'(1)E_{\mu'\backslash 2} = \mu(1)E_{\mu\backslash 2}$

. if $Divg(\lambda',\mu') < a$ then (I) is verified

. if $Divg(\lambda',\mu') = a$ then (λ',μ') or (μ',λ') being transformed by T_E, by validity of T_E , we obtain (I)

. if $Divg(\lambda',\mu') > a$

. either $\exists i \in [n_{\alpha,\beta}] / (\lambda',\mu') \in R_i \cup (R_i)^{-1}$

so $(\lambda',\mu') = (A'E_{\alpha'},B'E_{\beta'}Proj_E(<\gamma E_{\alpha'}>))^s$ with $(A'\alpha',B'\beta') = (\lambda,\mu)^s$
for $s \in \{-1,1\}$ and $(Proj_E(<\gamma\alpha'>,<\beta'>)^s$ is label of $Tree_G(\alpha,\beta)$
moreover $A' \equiv_{\bar{G}} B'\bar{\gamma}$ and by c) of lemma 6.2, we obtain
$Divg(A'E_{\alpha'}<\alpha'>,B'E_{\beta'}<\gamma E_{\alpha'}><\alpha'>) \geq Divg(\lambda',\mu') > a$
but $<\gamma E_{\alpha'}><\alpha'> = <\gamma\alpha'>$ so by d) of lemma 6.2,
$\min(Divg(A'E_{\alpha'}<\alpha'>,B'E_{\beta'}<\gamma\alpha'>) , Divg(B'E_{\beta'}<\gamma\alpha'>,B'E_{\beta'}<\beta'>)) \leq$
$Divg(A'E_{\alpha'}<\alpha'>,B'E_{\beta'}<\beta'>) = Divg(\lambda,\mu) = a$
therefore $Divg(B'E_{\beta'}<\gamma\alpha'>, B'E_{\beta'}<\beta'>) \leq a$
$(Proj_E(<\gamma\alpha'>),<\beta'>)^s$ being label of the tree and $Resp_G(\alpha,\beta) =$ "yes"
then we obtain $a > 1$
so, from b) of lemma 6.2, $Divg(<\gamma\alpha'>,<\beta'>) < a$
thus $Divg(Proj_E(<\gamma\alpha'>),<\beta'>) < a$ hence (I)

. or $(<\lambda\backslash 2>,<\lambda'\backslash 2>)$ and $(<\mu'\backslash 2>,<\mu\backslash 2>)$ are labels of $Tree_G(\alpha,\beta)$
but $\min(Divg(\lambda',\mu'),Divg(\lambda',\lambda),Divg(\mu',\mu)) \leq Divg(\lambda,\mu) = a$
and $Divg(\lambda',\mu') > a$ therefore
$\min(Divg(\lambda(1)E_{\lambda\backslash 2}<\lambda'\backslash 2>,\lambda(1)E_{\lambda\backslash 2}<\lambda\backslash 2>),$
$Divg(\mu(1)E_{\mu\backslash 2}<\mu'\backslash 2>,\mu(1)E_{\mu\backslash 2}<\mu\backslash 2>)) \leq a$
then as precedently $a > 1$ and by b) of lemma 6.2
$\min (Divg(<\lambda'\backslash 2>,<\lambda\backslash 2>),Divg(<\mu'\backslash 2>,<\mu\backslash 2>)) < a$ hence (I)

So, (I) is satisfied and by iterations of (I) from $\alpha \not\sim \beta$ with $Resp_G(\alpha,\beta)$ = "yes", we come a contradiction, so $\alpha \sim_G \beta$ which completes the proposition 8.2 □

Theorem 5 :

The equivalence problem for stateless deterministic pushdown automata (with acceptance for pushdown letters) is decidable and there exists a decision algorithm of polynomial complexity depending on the description length and the valuation of compared automata, or of exponential complexity if we only take account of the length of description.

Proof :

Remembering that $e_G = \max \{\tau_f(\alpha) / \alpha \in Im(G)\}$, $\ell_G = \max \{\tau_f(A) / A \in Dom(G)\}$
$|G| = \max \{|\alpha|/ \alpha \in Im(G)\}$, $\|G\| = \#Dom(G)$, $\Sigma_G = \{\alpha(1) \in \Sigma / \alpha \in Im(G)\}$
then we obtain $e_G \leq |G|.\ell_G$ and $\#G \leq \|G\|.\#\Sigma_G$

Let $n_{\alpha,\beta}$ be the last index of the sequence $(T_i,u_i,R_i)_{i\geq 0}$ of successive calling parameters of Const, with $T_0 = \{(\varepsilon,(Proj_E(\alpha),Proj_E(\beta)))\}$, $u_0 = \varepsilon$ and $R_0 = \emptyset$. Let us denote $S_{\alpha,\beta}$ the set of labels of $Tree_G(\alpha,\beta)$ having been transformed by T_E.

We have seen (proof of proposition 8.1) that :

$\forall u \in Dom(Tree_G(\alpha,\beta))$, $\tau_f(Tree_G(\alpha,\beta)(u)) \leq 2(\#G)^2 e_G + \tau_f(\alpha,\beta)$

To prove the theorem, we just have to show that the size of the equivalence tree $\#Tree_G(\alpha,\beta)$ admits a bound which polynomialy depends on ℓ_G, e_G, $|G|$, $\|G\|$, $\#\Sigma_G$, $\#G$.

Let us consider

$\mathcal{F}'' = Dom(S) \cup Im(S) / S = R_{\alpha,\beta} \cup T_E(S_{\alpha,\beta}) \cup \{(Proj_E(\alpha),Proj_E(\beta))\}$
$\mathcal{F}' = \{Proj_E(\lambda\downarrow R_i) / i \in \{0\} \cup [n_{\alpha,\beta}]$ and $\lambda \in \mathcal{F}'' \}$
$\mathcal{F} = \{\lambda\backslash j / \lambda \in \mathcal{F}'$ and $j \in \{0\} \cup [|\lambda|-1]\}$

Then $\#\mathcal{F}'' \leq 2(\#R_{\alpha,\beta} + \#\Sigma_G.\#S_{\alpha,\beta} + 1)$
but $\#R_{\alpha,\beta} \leq \#S_{\alpha,\beta} < 2(\#G)^2$
so $\#\mathcal{F}''$ is in $O((\#G)^2.\#\Sigma_G)$
furthermore $\#\mathcal{F}' \leq \#\mathcal{F}''.(\#R_{\alpha,\beta} + 1)$
so $\#\mathcal{F}'$ is in $O((\#G)^4.\#\Sigma_G)$
at last $\#\mathcal{F} \leq (\#\mathcal{F}').\max \{\tau_f(\Phi) / \Phi \in \mathcal{F}'\} = (\#\mathcal{F}').\max\{\tau_f(\Phi) / \Phi \in \mathcal{F}''\}$

thus $\#\mathcal{F}$ is in $O((\#G)^4.\#\Sigma_G.[(\#G)^2.e_G + \tau_f(\alpha,\beta)])$

in addition, we have the following properties :

$\mathcal{F}'' \subset \mathcal{F}' \subset \mathcal{F}$

$\forall \lambda \in \mathcal{F}$, $\forall j \in [|\lambda|-1]$, $\lambda\backslash j \in \mathcal{F}$

$\forall \lambda \in \mathcal{F}$, $\forall i \in [n_{\alpha,\beta}]$, $Proj_E(\lambda\downarrow R_i) \in \mathcal{F}$

hence $\forall i \in \{0\} \cup [n_{\alpha,\beta}]$, $\forall v \in Fl(T_i)$, $Simpl(T_i(v),R_i) \in \mathcal{F}\times\mathcal{F}$

in particular, $\forall i \in \{0\} \cup [n_{\alpha,\beta}]$, $Simpl(T_i(u_i),R_i) \in \mathcal{F}\times\mathcal{F}$

so $\#Tree_G(\alpha,\beta) < 2(\#\mathcal{F})^2$, that proves the theorem 5 □

Remark : this evaluation of the algorithm complexity is a roughly one.

We will be able to improve this algorithm such that, in the case of the root is an equivalent pair, the set of labels transformed by T_E of the equivalence tree is a self-proving system and all the leaves of the tree are reflexives. Then, it will become interesting to compute a bound of Divg.

Conclusion

What we must remember from this paper is that the considered equivalences are decidable because they are finitely generated and that the knowledge of the minimal generating systems of these equivalences enables us to define branching algorithms of least complexities. From [Co 83 b] and [Ha-Ha-Ye 79] 'studies, this method has been generalized to the equivalence of any classes of grammars and we obtain a general and sufficient condition to decide on grammars equivalence (to be publish). We could apply this condition to others existing equivalence algorithms, particularly for the dpda class with the real-time dpda class : [Oy-Ho-In 80] and especially [To 84].

Acknowledgment

I wish to thank Marie-France Escribe for translating this paper

I am grateful to Bruno Courcelle for his comments, to Philippe Darondeau for his careful reading of this paper, to Laurent Kott for his support.

References

Bu 73 P. Butzbach *Sur l'équivalence des grammaires simples* Actes des premières journées d'informatique théorique. Langages algébriques. Bonascre. pp 223-245

Ca 85 D. Caucal *Décidabilité de l'équivalence forte des R.P.S. monadiques* Rapport LITP. Thèse 3eme cycle. Université Paris 7

Ca 86 D. Caucal *Décidabilité de l'égalité des langages algébriques infinitaires simples* STACS 86 . LNCS 210 . pp 37-48

Co 74 B. Courcelle *Une forme canonique pour les grammaires déterministes* Rairo 1. pp 19-36

Co 78 B. Courcelle *A representation of trees by languages* TCS 6 & 7 . PP 255-279 & 25-55

Co 83 a B. Courcelle *An axiomatic approach to the KH algorithms* Math. Systems Theory 16. pp 191-231

Co 83 b B. Courcelle *Fundamental properties of infinite trees* TCS 25 . pp 95-169

Co-Vu 76 B. Courcelle, J. Vuillemin *Completeness result for the equivalence or recursive schemes* JCSS 12 . pp 179-197

Fr 77 E.P. Friedman *Equivalence problems for deterministic context-free languages and monadic recursion schemes* JCSS 14 . pp 344-359

Gi-Gr 66 S. Ginsburg, S.A. Greibach *Deterministic context-free languages* Information and control 9 . pp 602-648

Ha 78 M.A. Harrison *Introduction to formal language theory* Addison-Wesley

Ha-Ha Ye 79 M.A. Harrison, I.M. Havel, A. Yeduhaï *On equivalence of grammars through transformation trees* TCS 9 pp 191-231

Hu 80 G. Huet *Confluent reductions : abstract properties and applications to term rewriting systems* JACM 17 . pp 797-821

Ko-Ho 66 A.J. Korenjak, J.E. Hopcroft *Simple deterministic languages* Seventh Annual IEEE Switching and Automata Theory Conference . pp 36-46

Ol-Pn 77 T. Olshansky, A. Pnueli *A direct algorithm for checking equivalence of LL(k) grammars* TCS 4 pp 321-349

Oy-Ho 78 M. Oyamaguchi, N. Honda *The decidability of equivalence for deterministic stateless pushdown automata* Information Control 38 . pp 367-376

Oy-Ho In 80 M. Oyamaguchi, N. Honda, Y. Inagaki *The equivalence problem for real-time strict deterministic languages* Information Control 45 . pp 90-115

To 83 E. Tomita *A direct branching algorithm for checking equivalence of strict deterministic vs. LL(k) grammars* TCS 23 . pp 129-154

To 84 E. Tomita *An extended direct branching algorithm for checking equivalence of deterministic pushdown automata* TCS 32 . pp 87-120

Va 74 L.G. Valiant *The equivalence problem for deterministic finite-turn pushdown automata* Information Control 25 . pp 123-153

Va-Pa 75 L.G. Valiant, M.S. Paterson *Deterministic one-counter automata* JCSS 10 . pp 340-350

Wo 73 D. Wood *Some remarks on the KH algorithm for s-grammars* BIT 13 . pp 476-489

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique